TECHNISCHE UNIVERSITÄT DRESDEN

FAKULTÄT INFORMATIK INSTITUT FÜR SYSTEMARCHITEKTUR

Professur Rechnernetze

Diplomarbeit

zum Thema

Entwicklung und Umsetzung einer Methode zur

Genauigkeits- und Leistungsanalyse

von Netzwerkemulatoren

von Peter Büschel

Forschungsgebiet	Pervasive Computing and Collaboration
Forschungsprojekt	NESSEE
Studiengang	Informationssystemtechnik
Jahrgang	2008
Matrikelnummer	347 1993
Semester	WS 13/14

verantwortlicher Hochschullehrer

Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Betreut durch

Dipl.-Inf. Robert Lübke

Eingereicht am 31. Januar 2014, in Dresden

TECHNISCHE UNIVERSITÄT DRESDEN
Fakultät Informatik, Institut für Systemarchitektur, Lehrstuhl Rechnernetze

AUFGABENSTELLUNG FÜR DIE DIPLOMARBEIT

Name, Vorname:	Büschel, Peter		
Studiengang:	Informationssystemtechnik	Matrikelnummer:	3471993
Forschungsgebiet:	Pervasive Computing and Collaboration	Forschungsprojekt:	NESSEE
Betreuer:	Dipl. Inf. Robert Lübke	Externe(r) Betreuer:	-
Verantwortlicher Hochschullehrer:		*Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill*	
Beginn am:	01.07.2013	Einzureichen am:	31.12.2013
Thema:	Genauigkeits- und Leistungsanalysen von Netzwerkemulatoren		

ZIELSTELLUNG

Netzwerkemulation wird u.a. bei Experimenten mit Netzwerkanwendungen dafür eingesetzt, die in realen Netzwerken anzutreffenden Bedingungen, Parameter und Effekte nachzubilden. Es ist eine gewisse Genauigkeit der eingestellten Parameter (z.B. Paketverzögerung und -verlust) erforderlich, um eine Reproduzierbarkeit dieser Experimente zu gewährleisten. Sollen große Szenarien nachgebildet werden, spielt auch die Leistung des Emulators eine wichtige Rolle.
Das Ziel dieser Arbeit ist es, existierende und momentan in Forschung und Industrie eingesetzte Netzwerkemulatoren hinsichtlich dieser beiden Eigenschaften – Genauigkeit und Performance – eingehend zu untersuchen. Dazu sollen die an die Genauigkeits- und Leistungsmessungen gestellten Anforderungen ermittelt werden. Weiterhin soll ein entsprechendes Messkonzept entworfen, implementiert und evaluiert werden. Daraufhin sollen die Genauigkeits- und Leistungsmessungen mithilfe des entwickelten Messkonzepts für möglichst viele der existierenden Netzwerkemulatoren durchgeführt werden. Dabei sollen neben Software-basierten Netzwerkemulatoren für verschiedene Betriebssysteme (Windows, Unix, FreeBSD) auch Hardware-basierte Lösungen analysiert werden. Letztlich sollen die Ergebnisse der Messungen entsprechend ausgewertet werden.

SCHWERPUNKTE

- Grundlagen der Netzwerkemulation (Einsatz, Funktionsweise)
- Verwandte Arbeiten: Überblick existierender Software- und Hardware-basierter Lösungen
- Analyse der Anforderungen an Genauigkeits- und Leistungsmessungen
- Entwurf, Implementierung und Evaluierung eines Messkonzeptes
- Durchführung der Genauigkeits- und Leistungsmessungen mit möglichst vielen existierenden Netzwerkemulatoren, Auswertung der Ergebnisse

---

Verantwortlicher Hochschullehrer
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Vorsitzender Prüfungsausschuss IST
Prof. Dr.-Ing. habil. Martin Wollschläger

Für meine Liebsten,
Jessika und Eleonora
(Rückhalt und Freude).

Inhaltsverzeichnis

Abkürzungsverzeichnis . . . . . **v**

Symbolverzeichnis . . . . . **ix**

Abbildungsverzeichnis . . . . . **xi**

Tabellenverzeichnis . . . . . **xv**

1 Einführung **1**
1.1 Motivation 2
1.2 Aufgabenstellung 2

2 Nachbildung von Netzwerkverhalten **3**
2.1 Methoden zur Nachbildung 3
2.2 Taxonomiekonzept 5
2.3 Funktionsweisen 6
2.3.1 Architektur 7
2.3.2 Arbeitsbereich 8
2.3.3 Zeitgeber 8
2.3.4 Bedarf 9
2.3.5 Pfadvorgabe 9
2.4 Effektqualität/Netzwerkmetriken 11
2.4.1 Datenraten 11
2.4.2 Verzögerung 13
2.4.3 Paketeffekte 15
2.4.4 Mobilfunk 16
2.5 Skalierbarkeit 18
2.6 Zusammenfassung 19

3 Verwandte Arbeiten **23**
3.1 Auswahl der Emulatoren 23
3.2 Produktpublikationen und -dokumentationen 25
3.2.1 Linktropy 7500 pro 25
3.2.2 DummyNet 26
3.2.3 KauNet 30
3.2.4 ModelNet 33
3.2.5 NetEm 36
3.2.6 WANem 38

3.2.7 Imunes . . . 39
3.2.8 NetPath . . . 41
3.3 Vergleichsstudien . . . 44
3.3.1 Nussbaum, L. und Richard, O. (2009) . . . 44
3.3.2 Shaikh, J. und Minhas, T. et al. (2010) . . . 46
3.3.3 Minhas, T. und Shaikh, J. et al. (2011) . . . 48
3.4 Zusammenfassung . . . 49

4 Anforderungsanalyse . . . 53
4.1 Funktionale Anforderungen . . . 53
4.1.1 Verfahren . . . 53
4.1.2 Genauigkeits- und Leistungsmessungen . . . 55
4.1.3 Implementation . . . 56
4.1.4 Auswertung . . . 56
4.2 Qualitative Anforderungen . . . 56
4.3 Abgrenzungskriterien . . . 58
4.4 Zusammenfassung . . . 58

5 Konzeption . . . 61
5.1 Versuchsaufbau . . . 61
5.1.1 Hardwaretechnischer Aufbau . . . 61
5.1.2 Softwaretechnischer Aufbau . . . 63
5.2 Vorgehensweise . . . 69
5.2.1 Vorfelduntersuchungen . . . 70
5.2.2 Messungen . . . 71
5.2.3 Auswertung . . . 74
5.3 Zusammenfassung . . . 83

6 Umsetzung . . . 85
6.1 Versuchsaufbau . . . 85
6.1.1 Hardwaretechnischer Aufbau . . . 85
6.1.2 Softwaretechnischer Aufbau . . . 86
6.2 Vorgehensweise . . . 89
6.2.1 Vorfelduntersuchungen . . . 91
6.2.2 Messungen . . . 92
6.3 Zusammenfassung . . . 93

7 Validation . . . 95
7.1 Versuchsaufbau . . . 95
7.2 Paketgenerierung und -erfassung . . . 95
7.3 Fehlererkennung . . . 97
7.4 Zusammenfassung . . . 99

8 Durchführung **101**
8.1 Vorfelduntersuchungen 101
8.1.1 Grundverzögerungen 101
8.1.2 maximale verlustfreie Paketrate 102
8.1.3 Puffergröße 103
8.2 Messungen 104
8.2.1 Datenrate 105
8.2.2 Verzögerung 107
8.2.3 Paketeffekte 110

9 Zusammenfassung und Ausblick **115**

Anhang **119**
A.1 Nachweis Versuchsaufbau: Ping-antworten 120
A.2 Konfigurationsdatei für die Netzwerkeinstellungen 121
A.3 Konfigurationsdatei für die Angabe von Messparametern 122
A.4 Vorfelduntersuchungen: Grundverzögerungen 123
A.5 Messung: Datenrate 124
A.6 Messung: Verzögerungen 125
A.7 Messung: Neuordnung 126
A.8 Erklärung zum Urheberrecht 127

Bibliografie **129**
B.9 Literaturverzeichnis 129
B.10 Quellenverzeichnis 131

Abkürzungsverzeichnis

BER	Bit Error Rate (dt. Bitfehlerrate) - siehe Seite 16, 27, 34, 38–40, 42
BPS	Bits Per Second (dt. Bits pro Sekunde oder Datenrate). - siehe Seite 26, 77, 78
BTT	Bulk TCP Throughput (dt. Massendurchsatzrate), maximale Durchsatz bei TCP–Verbindungen - siehe Seite 13, 40, 44
CDF	Cumulative Distribution Function (dt. kumulierte Verteilungsfunktion) - siehe Seite 80, 107
CoDVwBMA	Coefficient of Delay Variation with Blocking Moving Average (dt. Variationskoeffizient der Verzögerungen mit blockierendem gleitendem Mittelwert) - siehe Seite 61, 81, 82, 107
CoTV	Coefficient of Throughput Variation (dt. Variationskoeffizient des Datendurchsatzes) - siehe Seite 46–48, 50, 56, 61, 81, 115
CoV	Coefficient of Variation (dt. Variationskoeffizient) - siehe Seite 81, 108
CT	Cross Traffic, Datenpakete fremder Anwendungen, die die Probepakete kreuzen. - siehe Seite 12, 27, 28, 42, 50, 77
FIFO	First In First Out (FIFO), Warteschlangenabarbeitung - siehe Seite 26, 27
GPS	Global Positioning System, unter Anderem zur Bereitstellung eines einheitlichen Zeitsystems - siehe Seite 37, 38
GUI	Graphical User Interface (dt. grafische Benutzeroberfläche) - siehe Seite 40, 56, 64
ICMP	Internet Control Message Protocol (dt. Internet-Kontrollnachrichtenprotokoll), für Diagnosezwecke - siehe Seite 45
IP	Internet Protocol - siehe Seite 26, 34
LEm	*Link emulator* (dt. Netzknotenemulator) - siehe Seite 5, 6, 18, 19, 25, 38, 49

MAD	mittleren absoluten Abweichungen (engl. mean absolute deviation (MAD)) - siehe Seite 56, 79
MLFFR	*maximum loss-free packet forwarding rate* (dt. maximale verlustfreie Paketrate) Quelle: [IHL07] - siehe Seite 42, 43, 50, 70, 71, 90, 91, 93, 101–103, 105
MTU	Maximum Transmission Unit, maximale erlaubte Größe eines Netzwerkpaket - siehe Seite 90
NTP	Network Time Protocol, zur Zeitsynchronisierung von Geräten im Netzwerk - siehe Seite 63
OWD	One Way Delay (dt. Verzögerung in eine Richtung der Verbindung) - siehe Seite 14, 27, 30, 34, 36–39, 42, 43, 47, 54, 61, 62, 67, 69–71, 77, 81, 91–93, 101, 103
PCI	Peripheral Component Interconnect; Bussystem, welches Hardware-Steckkarten mit dem Rechner verbindet - siehe Seite 85, 86, 91, 94
PCIe	Peripheral Component Interconnect Express; Bussystem, welches Hardware-Steckkarten mit dem Rechner verbindet - siehe Seite 85, 86, 94
PDF	Probability Distribution Function (dt. Verteilungsfunktion) - siehe Seite 80
PDR	Packet Duplication Rate (dt. Dopplungsrate, von Datenpaketen) - siehe Seite 15, 16, 27, 38, 39, 42, 101
PLR	Packet Loss Rate (dt. Verlustrate) - siehe Seite 15, 27, 30, 34, 36, 38, 39, 58, 101
PPS	Packets Per Second (dt. Pakete pro Sekunde oder Paketrate) - siehe Seite 29, 30, 50, 77, 78, 99, 105, 112
PPT	Per-packet Processing Time (dt. Verarbeitungszeit je Paket) - siehe Seite 29, 30, 50
PRR	Paket Reordering Rate (dt. Paketneuordnungsrate) - siehe Seite 16, 27, 30, 34, 36, 38, 39, 101
RR	Retransmission Rate (dt. Wiederholungsrate), von Datenpaketen - siehe Seite 15
RTD	Round Trip Delay (dt. Umlaufverzögerung), gleichbedeutend mit der *Round Trip Time* (RTT) - siehe Seite 14, 41, 44, 45, 63
RTT	Round Trip Time (dt. Umlaufzeit) eines Netzwerkpaket vom Sender über den Empfänger zurück zum Sender. - siehe Seite vi

TCP	Transmission Control Protocol (dt. Übertragungssteuerungsprotokoll) - siehe Seite 13, 26, 27, 37, 39–41, 43, 45, 50
UDP	User Datagram Protocol - siehe Seite 27, 29
VB	*virtuelle Betriebsmittel* - siehe Seite 19
VM	*virtual machine* (dt. virtueller Rechner) - siehe Seite 19, 20, 23, 33, 61
VNE	*virtual network emulator* (dt. Emulator basierend auf virtuellen Netzwerken) - siehe Seite 5, 6, 18, 19, 25
WAN	Wide Area Network (dt. Weitverkehrsnetz) - siehe Seite 25, 38

Symbolverzeichnis

A	verfügbare Datenrate in einem festen Zeitintervall 12, 13, 28, 78
C	maximale Datenrate 11–13, 25, 27, 39, 42, 101
D_P^j	Ausbreitungsverzögerung (engl. propagation delay) 14
D_Q^j	Warteschlangenverzögerung (engl. queuing delay) 14, 15
D_T^j	Übertragungsverzögerung (engl. transmission delay) 14
H	Anzahl der Teilabschnitte (engl. Hops) eines Verbindungspfades 35, 89
Kr	Realer Netzwerkknoten. 7, 18
Kv	Virtueller Netzwerkknoten. 7, 19
L	Paketgröße bzw. Anzahl Bits eines Pakets 28, 67
P	Verbindungspfad 11, 12
R_x	Empfänger 14
T_x	Sender 14
ΔOWD^j	Verzögerungsvariation des j-ten Paketes 14, 15, 25, 27, 37–39, 43, 67, 74, 77, 81
σ	Standardabweichung; Maß für die Streuung um den Erwartungswert einer Verteilung. 79
i	i-te Teilabschnitt eines Verbindungspfades 11
j	j-te Paket in einem Paketstrom. 14
u_i	Auslastung des i-te Teilabschnitt eines Verbindungspfades 12, 13
$x_{0.5}$	Zentralwert oder Median einer geordneten Liste.

79, 80, 92

x_D Modalwert oder Modus; Häufigster Wert in einer Verteilung.
79, 80

Abbildungsverzeichnis

Abb. 2.1 Wasserfallmodell der Softwareentwicklung mit Einsatzmöglichkeiten der drei Versuchsmethoden. . . . 3
Abb. 2.2 Hauptkategorien der entworfenen Taxonomie mit Angaben zu feststehen bzw. messbaren Eigenschaften. . . . 6
Abb. 2.3 Kategorie *Funktionsweise* mit den fünf zugehörigen Unterkategorien. . . . 7
Abb. 2.4 Pfad P mit den Teilabschnitten von $i = 1$ bis $i = H$ (Quelle: [Büs13]) . . . 11
Abb. 2.5 Kategorie *Effektqualität* mit den verwendeten Metriken. . . . 12
Abb. 2.6 P mit $H = 3$ und $C = C_1$ (Quelle: [Büs13]) . . . 13
Abb. 2.7 Definition der verschiedenen Raten. (Quelle: [Büs13]) . . . 13
Abb. 2.8 Schema der Verzögerungsphasen bei OWD. (Quelle: [Büs13]) . . . 14
Abb. 2.9 Verzögerungsvariationen. (Quelle: [Büs13]) . . . 15
Abb. 2.10 Kategorie *Skalierbarkeit* und deren Einflussfaktoren. . . . 18

Abb. 3.1 Analyse der Suchanfragen zwischen 2004 und 2013 für bekannte Emulatoren. (Quelle: [13f]) . . . 24
Abb. 3.2 Anzahl der Zitationen für ausgewählter Emulatoren. (Quelle: [13e]) . . . 24
Abb. 3.3 Weboberfläche des „Linktropy 7500 pro“ Hardware-Emulators. . . . 26
Abb. 3.4 Funktionsprinzip von DummyNet (nach [Riz97]) . . . 27
Abb. 3.5 Funktionsprinzip von DummyNet mit der Erweiterung durch ipfw (nach [CR10]) . . . 28
Abb. 3.6 Oben: die typische Messung der Paketrate durch die gesamte Paketverarbeitung des Betriebssystems. Unten: die verwendete Messmethode, um die Paketklassifikation und Emulation gesondert betrachten zu können. (Quelle: [CR10]) . . . 30
Abb. 3.7 (a) Zeit- und (b) datengesteuerte Auswahl bestimmter Pakete durch KauNet anhand des Musters „0101“ (Quelle: [Hur+12]) . . . 31
Abb. 3.8 Funktionsprinzip von KauNet mit dem Mustergenerator `patt_gen` . . . 31
Abb. 3.9 Die drei Versuchsumgebungen für die Analysen von KauNet. . . . 32
Abb. 3.10 Schematischer Aufbau von ModelNet am Beispiel mit drei *virtual edge nodes* (VN) auf zwei Realen Rechner und einem *Core Router Cluster* mit drei Rechnern. VN 1 und VN 2 verbinden sich parallel über teilweise unterschiedliche Pipes mit VN 3. . . . 34
Abb. 3.11 Baumartige Struktur der *qdisc*-Warteschlangen unter Linux. . . . 36
Abb. 3.12 Schema der Paketverarbeitung durch NetEm. . . . 37
Abb. 3.13 Leichtgewichtige Virtualisierung unter Imunes (Quelle: [PM06]) . . . 40
Abb. 3.14 Versuchsaufbau in [NR09]. . . . 44

Abb. 3.15 Messergebnisse: Einfluss der Taktrate auf die Genauigkeit bei der Emulation einer Paketverzögerung von 10 *ms*. (rechts: Häufigkeitsverteilung, Quelle: [NR09]) 45
Abb. 3.16 Versuchsaufbau in [Sha+10] angelehnt an die DPMI. 47
Abb. 3.17 Beispiel für die Visualisierung der Häufigkeitsverteilung und der verschiedenen CoTV. (links: Häufigkeitsverteilung, Quelle: [Sha+10]) 48

Abb. 4.1 Grundlegende Vorgehensweise 54
Abb. 4.2 Grundlegende Anforderungen an den Versuchsaufbau 54
Abb. 4.3 Ein- und Ausgabemöglichkeit der geforderten Endergebnisse. 55

Abb. 5.1 Erster Ansatz des eigenen Versuchsaufbaus. 62
Abb. 5.2 Angepasster Ansatz des eigenen Versuchsaufbaus mit zwei Messpunkten. . 63
Abb. 5.3 Hardware-technischer Versuchsaufbau mit zusammengelegtem Sender und Empfänger. 64
Abb. 5.4 Softwaretechnischer Versuchsaufbau. 65
Abb. 5.5 Einteilung der Versuchsparameter in Probeströme und zugehörige Probedurchläufe. 66
Abb. 5.6 Sequenzdiagramm der Versuchssteuerung bzgl. der Startreihenfolge der verschiedenen Grundkomponenten. 67
Abb. 5.7 Die Paketgenerierung in der vorangegangenen Arbeit [Büs13]. 68
Abb. 5.8 Die verbesserte Paketgenerierung in dieser Arbeit. 69
Abb. 5.9 Allgemeines Schema der Parameterverknüpfungen für die Messungen. . . . 73
Abb. 5.10 Beispiel für die Analyse von Paketverlusten und -dopplungen. 76
Abb. 5.11 Pseudocode: Bestimmung der Anzahl und Paketnummern von neu geordneten Paketen. 77
Abb. 5.12 Beispiel für das Vorgehen und die Berechnung der einzelnen OWD-Werte. 78
Abb. 5.13 Beispiel für die Ermittlung des Median einer Liste mit gerader Anzahl an Elementen. 80
Abb. 5.14 Schematischer Verlauf von links-, rechtsschiefen sowie symmetrischen Verteilungen. Oben: PDF ,Unten: CDF 80
Abb. 5.15 Berechnung der Durchschnittlichen ΔOWD für das jeweilige Zeitintervall ΔT auf der Empfängerseite. 82
Abb. 5.16 Visualisierung der berechneten CoDVwBMA für den Sender und Empfänger unter Berücksichtigung der Zeitintervalle $\Delta T = \{10\ ms, 20\ ms, \ldots, 60\ ms\}$. . . . 83
Abb. 5.17 Überblick über den Hard- und Software-technischen Versuchsaufbau inklusive der Versuchskomponenten sowie -parameter. 84

Abb. 6.1 Fotografie des realen Versuchsaufbau. Oben: Vergrößerte Aufnahme der unterschiedlichen Netzwerkkarten. 87
Abb. 6.2 Netzwerkeinstellungen des Versuchsaufbaus. 88
Abb. 6.3 Beispiel für die Prozessgenerierung der Paketerfassung sowie das Zusammenwirken von Python und einem C-Erweiterungsmodul. 89
Abb. 6.4 Vereinfachte Ausgabe des Befehl `vmstat -i` zur Analyse von gemeinsam genutzten Interrupts. 91

Abb. 7.1 Messergebnisse der erfassten Paketabstände durch WIRESHARK und der eigenen Implementation auf der Senderseite (a)–(c) sowie Empfängerseite (d)–(e) als Boxplot für die verschiedenen vorgegebenen Paketabstände und -größen. . . 98

Abb. 8.1 Box-Diagramme für die gemessenen Grundverzögerungen unter Verwendung der verschiedenen Stromeigenschaften am Beispiel von FREEBSD 7.3 (KAUNET). 102
Abb. 8.2 Gemessene Verzögerungswerte zur Ermittlung der MLFFR am Beispiel von FREEBSD 7.3 (KAUNET). 104
Abb. 8.3 Auszug aus den Messergebnissen zur Betrachtung der Genauigkeit bei der Umsetzung von verschiedenen vorgegebenen Datenraten für einen einzelnen Pfadabschnitt. Die Paketgröße beträgt bei allen Darstellungen 512 Byte. 106
Abb. 8.4 Relative Abweichungen der gemessenen Datenraten bei unterschiedlichen Stromeigenschaften und einem einzelnen Pfadabschnitt. Die vermeidlich fehlenden Balken in der Abbildung kommen dadurch zustande, dass mit der jeweiligen Kombination von Paketgröße sowie -abstand nicht die geforderte Datenrate erreicht wird. Vorgabe: 7200 KBit/s 107
Abb. 8.5 Auszug aus den berechneten CoDVwBMA-Werten zur Ermittlung der Wertstabilität des jeweiligen Emulators während der Messung der Verzögerungen. Vorgabe: 1 ms 108
Abb. 8.6 Gemessene Verzögerungen mit dem Emulator DUMMYNET als Box-Diagramm für die verschiedenen Paketlängen und -raten bei 10 Pfadabschnitten. Vorgabe: 10 ms 110
Abb. 8.7 Auszug aus den Messergebnissen zur Betrachtung der Genauigkeit bei der Umsetzung von 0,001 % Paketverlust für einen einzelnen Pfadabschnitt. Die Box-Diagramme von KAUNET erscheinen wegen der fehlenden Abweichung als einzelne Striche mit einem Stern (Durchschnittswert) in der Mitte. 110
Abb. 8.8 Erfasste Paketverluste als Box-Diagramme für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt. Auch hier erscheinen die Box-Diagramme von KAUNET wegen der fehlenden Abweichung als einzelne Striche mit einem Stern in der Mitte. 111
Abb. 8.9 Erfasste Paketverluste als Box-Diagramme für die verschiedenen Paketlängen sowie -raten bei 10 Pfadabschnitten und mit Dummynet als Emulator. Der Erwartungswert liegt bei 9,6 Verlusten. 112
Abb. 8.10 Erfasste Paketneuordnungen als Box-Diagramme für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt. 113
Abb. 8.11 Erfasste Paketdopplungen als Box-Diagramme für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt. 114

Abb. A.1 Grundverzögerungen als Verteilungsfunktion des jeweiligen Emulators für die verschiedenen Paketlängen und -raten. 123
Abb. A.2 Gemessene Datenraten bei unterschiedlichen Stromeigenschaften und einem einzelnen Pfadabschnitt. Vorgabe: 200 KBit/s 124
Abb. A.3 Gemessene Datenraten bei unterschiedlichen Stromeigenschaften und 10 Pfadabschnitten. Vorgabe: 200 KBit/s 124

Abb. A.4 Verzögerungen als Verteilungsfunktion des jeweiligen Emulators für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt. Vorgabe: 1 ms 125
Abb. A.5 Erfasste Neuordnungen für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt. Vorgabe: 1 % 126

Tabellenverzeichnis

Tab. 2.1 Beispiel für ein Bewertungsschema einiger ausgewählter *feststehenden Eigenschaften* der Klassen *Funktionsweise* und *Skalierbarkeit* anhand der genannten Vor- und Nachteile. (Bedeutung: „+“-positiver, „−“-negativer, „*o*“-mittelmäßiger und „ “-kein Zusammenhang) 21

Tab. 3.1 Angaben zur einstellbaren Granularität und Leistung des „Linktropy 7500 pro“ 27
Tab. 3.2 *timing*: Grad der Beeinträchtigung durch die genannten Faktoren. (Quelle: [CR10]) 29
Tab. 3.3 Produktangaben zur möglichen Granularität und Leistung von DummyNet (nach [13w]) 30
Tab. 3.4 Angaben zur Granularität und Leistung von KauNet 34
Tab. 3.5 Produktangaben zur Granularität und Leistung von ModelNet 36
Tab. 3.6 Angaben zur Granularität von NetEm (nach [13q]) 38
Tab. 3.7 Produktangaben zur Granularität und Leistung von WANem (nach [KN08]) 39
Tab. 3.8 Produktangaben zur Granularität von Imunes (nach [Uni12]) 42
Tab. 3.9 Angaben zur Granularität und Leistung von NetPath (nach [Aga05]) . . . 43
Tab. 3.10 Übersicht der feststehende Eigenschaften der ausgewählten Emulatoren. . . 51

Tab. 4.1 Übersicht Teilziele 57
Tab. 4.2 Übersicht qualitative Anforderungen 58
Tab. 4.3 Übersicht funktionale Anforderungen 59

Tab. 5.1 Allgemeine Probestromeigenschaften je Emulationsvorgabe für die einzelnen Bereiche bei der Untersuchung der Granularität und des Wertebereiches. Pro Vorgabe werden dabei mehrere Paketabstände und -größen untersucht. 74
Tab. 5.2 Beispielwerte für die Betrachtung der Wertstabilität mit Hilfe von CoDVwBMA. 81
Tab. 5.3 Ergebnisse der Mittelwertberechnungen für die Betrachtung der Wertstabilität mit Hilfe von CoDVwBMA. 83

Tab. 6.1 Hardwarekonfiguration der Versuchsrechner (vgl. Abb. 5.3). 85
Tab. 6.2 Allgemeine Ergebnismatrix und gleichzeitig Übersicht über die verwendeten Paketgrößen (a1)–(a4) und -abstände (b1)–(b3) (vgl. Abb. 5.9). 90
Tab. 6.3 Übersicht über die resultierenden Paket- sowie Datenraten bei der Kombination aus den Werten für (a1)–(a4) und (b1)–(b3) (vgl. Abb. 5.9). 90
Tab. 6.4 Übersicht der verwendeten Messparameter zur Ermittlung der Granularität und des Wertebereiches. 94

Tab. 7.1 Ausgabe von Wireshark für den Nachweis der Funktionalität des Versuchsaufbaus. 96

Tab. 8.1 Übersicht über die Funktionsmöglichkeiten der ausgewählten Emulatoren in Bezug auf die zu untersuchenden Netzwerkmetriken. 101
Tab. 8.2 Durchschnittliche Grundverzögerungen (in μs) des Versuchsaufbau für die jeweilige Paketstromeigenschaft unter FreeBSD 7.3 (KauNet). 102
Tab. 8.3 Mittelwerte der erfassten Grundverzögerungen (in μs) für den jeweiligen Emulator unter Verwendung von verschiedenen Paketgrößen und -abständen. . 103
Tab. 8.4 Durchschnittliche Verzögerungen (links) und Datenraten (rechts) für die jeweiligen Paketstromeigenschaften unter Verwendung von KauNet mit einer Datenratenbeschränkung von 1 MBit/s. 103
Tab. 8.5 Berechnete Puffergröße des Emulators KauNet für die jeweilige Paketstromeigenschaft. 104
Tab. 8.6 Puffergröße des jeweiligen Emulators. 104
Tab. 8.7 Abweichungen der mittleren emulierten Verzögerung bei verschiedenen Stromeigenschaften des jeweiligen Emulators von der Vorgabe. Pfadlänge: 1 . . 109

1 Einführung

Die Anzahl der Entwicklungen im Bereich der Netzwerkprotokolle und verteilten Anwendungen steigt kontinuierlich mit zunehmender Vernetzung des Alltags an.

Das Bundesamt für Sicherheit (BSI) veröffentlichte beispielsweise eine Trendstudie [Bro03], die für das Jahr 2013 eine durchgehend digitalisierte und automatisierte Wertschöpfungskette voraussagt. Ein Szenario daraus beschreibt wie der Wecker eines Geschäftsmannes mit dessen Online-Terminkalender vernetzt ist und ein Taxi zur Videokonferenz bestellt, in welcher die Teilnehmer aus verschieden Ländern gemeinsam Produktentwürfe diskutieren und deren Produktion anstoßen.

Das Beispielszenario zeigt indirekt auch die Gefahren auf, sobald eine Anwendung vom erwarteten Verhalten abweicht. Diese Fehlverhalten können zum Beispiel eine falsche Weckzeit oder aber auch die Weiterleitung eines bereits verworfenen Entwurfs an die Produktion beinhalten. Dadurch entstehen finanzielle Einbuße oder, beispielsweise im Falle von Fehlproduktionen im Verkehrsbereich, sogar Gefahren für Menschenleben.

Eine der Ursachen für solche Fehlverhalten sind Einflüsse auf die Datenpakete während der Übertragung, welche zumeist nur in Ausnahmesituationen auftreten. Anwendungen sollten im optimalen Fall darauf reagieren können und entsprechende Maßnahmen ergreifen. Damit die Entwickler einer solchen Anwendung Fehlerfälle überprüfen können, ist eine umfassende Validierung nötig, welche jedoch stark von der Testumgebung bzw. dem Testverfahren abhängig ist.

Das Verhalten von Protokollen und Anwendungen lässt sich in Modellen simulieren oder kann in realen Netzwerkumgebungen experimentell untersucht werden. Beide Vorgehen einzeln betrachtet bergen jedoch Nachteile in sich, die im 2. Kapitel näher erläutert werden.

Ein weitere Methode ist das Nachstellen von Netzwerkeffekten auf die Daten einer Übertragung mittels Emulation. Mit dieser ist es möglich die Wechselwirkungen zwischen einer Netzwerkanwendung und einem Verbindungspfad zu untersuchen, dessen Eigenschaften durch eine gezielte Beeinflussung weitestgehend vorgegeben werden können. Durch die Emulation entstehen so kontrollierbare und reproduzierbare Testabläufe.

Die Verwendung eines Emulators beeinflusst jedoch zusätzlich die Testumgebung und verursacht Approximationen in den Testergebnissen. Um diese Umstände bei den Beobachtungen berücksichtigen zu können, sollten bereits im Vorfeld der Testabläufe Genauigkeits- und Leistungsanalysen der verwendeten Netzwerkemulatoren durchgeführt werden.

Die Entwickler der einzelnen Emulatoren bieten zumeist solche Analysen für ihr Produkt an, jedoch unterscheiden sich diese Messverfahren untereinander und deren Ergebnisse werden nur auf bestimmte Nachweise ausgerichtet. Vergleiche mit anderen Emulatoren werden selten beschrieben oder komplett weggelassen.

1.1 Motivation

Die Motivation zu dieser Arbeit ergibt sich daraus, ein einheitliches Messverfahren zu schaffen, welches die Möglichkeit bietet Genauigkeits- sowie Leistungsanalysen automatisiert für möglichst viele Emulatoren durchzuführen. Dadurch ergeben sich Vergleichsmöglichkeiten zwischen den verschieden Netzwerkemulatoren und die Eignung des jeweiligen Emulators kann für vordefinierte Testszenarien festgestellt werden.

Die Genauigkeitsanalyse bezieht sich auf die eingestellten Netzwerkparameter, welche im Abschnitt 2.4 kurz umrissen werden. Die Messverfahren dazu bauen auf einer vorhergehenden Arbeit [Büs13] auf, in welcher ein integriertes Netzwerkanalysewerkzeug namens Nora entwickelt wurde.

Die Leistungsanalyse betrachtet zum einen den Funktionsumfang und zum anderen die mögliche Belastbarkeit sowie Skalierbarkeit des jeweiligen Emulators. Die genaue Aufschlüsslung der Teilbereiche ist in Kapitel 4 zur Anforderungsanalyse zu finden.

1.2 Aufgabenstellung

Es werden mit dieser Arbeit drei Ziele verfolgt. Das erste Ziel soll es sein eine Klassifizierung für die verschiedenen Emulatoren zu entwerfen, um mögliche Gemeinsamkeiten zwischen den Emulatoren für ein einheitliches Messkonzept zu finden und um die Konzeption in bestimmte Untersuchungsbereiche einteilen zu können. Dazu sollen zunächst grundlegende Funktionsweisen und Einsatzgebiete von Emulatoren erläutert werden.

Das zweite Ziel ist es, ein Messkonzept zu erstellen und umzusetzen, welches es ermöglicht die Genauigkeits- und Leistungsanalysen durchzuführen. Diese Lösung umfasst dabei den Versuchsaufbau und eine Anwendung zur Wertermittlung sowie Analyse. Hierbei sollen die Messkonzepte von verwandten Arbeiten mit betrachtet und eine Analyse der Anforderung an ein solches Konzept vorgenommen werden.

Die Durchführung der Analysen bei verschiedenen Emulatoren mit Hilfe des Messkonzepts bzw. dessen Umsetzung ist das dritte Ziel dieser Arbeit.

2 Nachbildung von Netzwerkverhalten

Testumgebungen sind ein wichtiges Hilfsmittel für die Entwicklung und Validierung von Anwendungen. Durch diese ist möglich, bereits im Vorfeld Anomalien zu erkennen. Auch verhelfen Testumgebungen den Entwicklern zu einem distanzierterem Blick auf ihr Projekt.

Oft arbeitet ein Entwickler lediglich an einem Bereich der Software und erst in der Testphase werden alle Teile zusammengefügt. Daher ist es entscheidend adäquate Testmethoden und Versuchsumgebungen zur Verfügung zu haben. Je nach Entwicklungsstufe des Softwareprojekts werden dazu oft Versuche mittels *Realumgebung*, *Simulation* oder *Emulation* durchgeführt. In Abbildung 2.1 wird beispielhaft die Einsatzmöglichkeit der jeweiligen Versuchsmethode am Wasserfallmodell gezeigt. Eine Kombination aus den beiden letztgenannten Versuchsmethoden

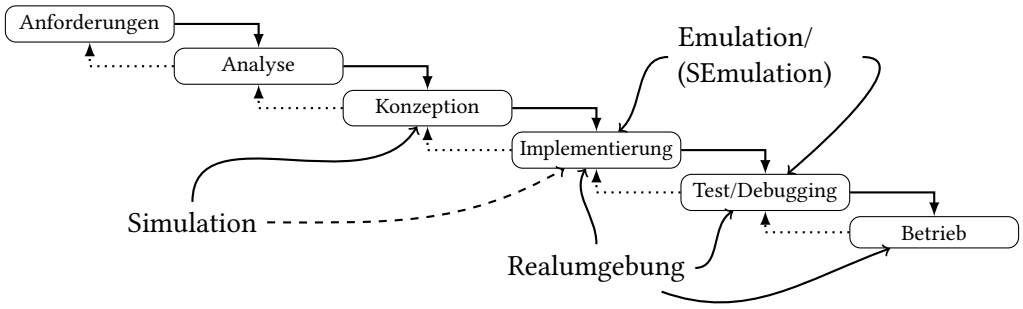


Abbildung 2.1: Wasserfallmodell der Softwareentwicklung mit Einsatzmöglichkeiten der drei Versuchsmethoden.

wird ebenfalls für bestimmte Testszenario verwendet und kann als *SEmulation* bezeichnet werden. Dieser Begriff stammt ursprünglich aus dem Bereich der Schaltkreis- und Systementwicklung und wird in dieser Arbeit für eine mögliche Methode zur Nachbildung von Rechnernetzen verwendet.

Der Fokus dieser Arbeit liegt auf der Genauigkeits- und Leistungsanalyse von Netzwerkemulationen bzw. deren unterschiedlichen Implementationen in Form von Emulatoren.

Bevor jedoch weiter auf Netzwerkemulatoren eingegangen wird, werden zunächst die genannten Methoden für eine begriffliche Abgrenzung kurz umrissen.

2.1 Methoden zur Nachbildung

Simulation werden in der Regel als Vorhersage- und Managementwerkzeug für Netzwerke benutzt. Dabei existieren zwei Ansätze Simulationen durchzuführen[Luc+03]. Der erste Ansatz beruht auf mathematischen Modellen die Netzwerke und deren Einflüsse abstrahieren. Paketströme werden mit Hilfe von Differentialgleichungen nachgebildet. Die Ergebnisse dieser Gleichungen geben u. a. die mittlere Warteschlangenverzögerung an. Daher arbeitet dieser Ansatz

auf einer höheren Netzwerkebene. So können beispielsweise verschiedene Ausbreitungsmodelle für kabellose Netzwerke getestet werden, ohne den tatsächlichen Aufbau von Antennen[IHL07]. Diese analytischen Methoden bringen schneller Ergebnisse hervor als der zweite Ansatz, sind jedoch nicht so präzise wie dieser.

Der zweite Ansatz ist Ereignisgetrieben[SU03] und versucht Vorhersagen auf der Ebene von Datenpaket zu treffen. Dieser gestattet dem Entwickler sich auf die kritischen Ereignisse zu konzentrieren und verschiedene Konzepte zu untersuchen. Dadurch können präzisere Ergebnisse gefunden werden als beim analytischen Ansatz.

Ein Nachteil bei beiden Ansätzen ist jedoch, dass die Ergebnisse stark von der Güte des benutzten Modelles abhängt. Nicht alle Faktoren in einem Netzwerk sind modellierbar, da sie oft dem echtem Zufall unterliegen. Auch steigt mit zunehmender Komplexität simulierter Netzwerke die Berechnungsdauer[HK05] und es können keine Experimente in Echtzeit durchgeführt werden. Ein simulierte Sekunde kann so eine reale Stunde dauern. Ein anderer Nachteil ist, dass die zu testende Anwendung erst an ein bestimmtes Modell angepasst werden muss und so zusätzliche Fehlerquellen entstehen.

Deshalb werden Simulationen oft zusammen mit Realumgebungen betrachtet.

Realumgebung werden in der Implementationsphase für die Untersuchungen des Programmverhaltens unter realen Bedingungen benutzt. Dazu wird reale Hardware sowie Software benutzt und zumeist die Zielplattform nachkonstruiert. Im Netzwerkbereich können solche Plattformen jedoch sehr kostenintensiv werden, je komplexer die Zieltopologie ausfällt. Teilweise ist es auch von vornherein nicht möglich ein reales Netzwerk für eine beispielsweise neue Übertragungstechnik aufzubauen. Im Satellitenbereich müssten erst entsprechende Satelliten ins All geschickt werden.

Nachteilig wirkt sich auch die Tatsache aus, dass bestimmte Netzwerkeffekte nur unregelmäßig auftreten. Eine kontrollierbare Testumgebung mit reproduzierbaren Ergebnissen ist daher mit Realumgebungen nicht gegeben.

Emulation kann als Verknüpfung von Realumgebung und Simulation angesehen werden. Allgemein bildet eine Emulation die Funktionen und das Verhalten eines Systems nach, ohne dessen Abhängigkeit von bestimmten Ressourcen.

Die Netzwerkemulation erlaubt die Nachahmung von Effekten in realen Verbindungspfaden bis hin zur Nachbildung von Netzwerktopologien[Gra+08]. Für die Emulation einer Topologie wird eine weitere Abstraktionsebene zwischen realen Netzwerk und Emulator hinzugefügt. In dieser Ebene es möglich virtuelle Netzwerkelemente zu erstellen, welche in der Lage sind mit realen Elementen zu kommunizieren. Dadurch kann nahezu in Echtzeit das Gesamtverhalten einer Anwendung in einer vorgegebenen Topologie mit gezielten Pfadmanipulationen untersucht werden.

Metriken wie Speicherverbrauch, Prozessorauslastung oder die Latenz bei der Behandlung von Datenpaketen können nicht durch eine Simulation gemessen werden. Um verschiedene Bandbreiten testen zu können, müssten bei Realumgebungen mehrere Verbindungswege zur Verfügung stehen. Die Emulation dagegen gestattet einen Überblick über die verwendeten Ressourcen und erlaubt eine einfache Einstellungsmöglichkeit der gewünschten Netzwerkeigenschaft. Daher eignen sich Emulatoren besonders für die Test- und Debuggingphase.

SEmulation ist die Kombination aus Simulation und Emulation. Wobei zwei mögliche Varianten zur Kombination existieren. In der erste Variante wird zunächst eine Simulation bestimmter Ereignisse durchgeführt, um daraus die Eigenschaften der Zielplattform zu berechnen. Die Ergebnisse dienen im nächsten Schritt einem Emulator als Grundlage für dessen Versuchsdurchführung. Beispielsweise kann so die Empfangsqualität eines Satellitensignals für verschiedene Gebiete simuliert und die berechneten Werte anschließend in einer Testumgebung emuliert werden.

Bei der anderen Variante agieren simulierte Netzwerkkomponenten in Echtzeit mit realen Netzelementen, die wiederum emulierte Eigenschaften besitzen. Die simulierten Komponenten können dabei komplexe Bereiche eines Netzwerkes abstrakt darstellen. Die Simulation beruht auf möglichst realitätsnahen Modellen, die im vorgegebenen Maß bestimmte Ereignisse auslöst. Reale Datenpakete treffen so auf simulierte Ereignisse. In der englischen Literatur wird dafür die Bezeichnung *Real-time Discrete event Driven Simulation*[SU03] benutzt.

Bei der Kombination aus Simulation und Emulation werden Vorteile aus beiden Testverfahren ausgenutzt. Allerdings überwiegen die Nachteile der Simulation je komplexer und detaillierter eine Netzwerkkomponente wird. Daher kann die *SEmulation* nur bedingt eingesetzt werden.

2.2 Taxonomiekonzept

Nach diesem kurzen Überblick über die verschieden Versuchsmethoden, lässt sich zusammenfassen, dass Emulationen überall dort eingesetzt werden, wo eine kontrollierbare Testumgebung benötigt wird, die in Echtzeit reproduzierbare Versuchsergebnisse liefert. Wobei der Versuchsaufbau aus realen oder virtuellen Netzwerkkomponenten bestehen kann.

Die Umsetzung der Netzwerkemulation geschieht mit Hilfe von Emulatoren. Bei der Versuchsplanung sollte berücksichtigt werden, dass die verschiedenen Emulatoren mit unterschiedlichen Schwerpunkten konzipiert wurden. Um die Wahl des richtigen Emulators für ein bestimmtes Testszenario zu erleichtern, bietet es sich an, die vorhandenen Emulatoren in ein Klassifikationsschema einzuteilen.

In der Literatur wird oft zwischen einfachen Netzknotenemulatoren (engl. *Network Link Emulator* (NLE)) und Emulatoren unterschieden, welche in der Lage sind virtuellen Netzwerke nachzubilden (engl. *Virtual Network Emulator* (VNE)) (vgl. [NR09],). Diese feststehenden Eigenschaften sind jedoch nur ein Unterscheidungspunkt, wenn es um die Auswahl des Emulators für ein bestimmtes Testszenario geht. Wichtig sind, neben weiteren festen Merkmalen wie der Mindestanforderung, auch die messbare Eigenschaften, welche zum Beispiel Auskunft über die Genauigkeit des Emulators geben.

Daher werden in diesem Kapitel weitere Unterscheidungsmerkmale betrachtet, die beide Arten von Eigenschaften berücksichtigen. Gleichzeitig entsteht so eine Taxonomie für Netzwerkemulatoren, die sich zu einem Teil aus den Produkteigenschaften und zum anderen Teil aus Genauigkeits- und Leistungsanalysen zusammensetzt. Die Taxonomie wird anschließend für die Anforderungsanalyse in Kapitel 4 und in der darauffolgenden Konzeption benötigt. Darauf aufbauend lässt sich so ein Schema für die Bewertung und Einsatzmöglichkeiten des jeweiligen Emulators ableiten.

In dieser Arbeit werden dafür, wie in Abb. 2.2 dargestellt, die Emulatoren aufgrund ihrer *Funktionsweise, Effektqualität* und *Skalierbarkeit* klassifiziert. Die feststehenden Merkmale werden dabei aus den Dokumentationen des jeweiligen Emulators entnommen. Die messbaren

Eigenschaften dagegen sind Gegenstand der Genauigkeits- und Leistungsanalysen zu dieser Arbeit.

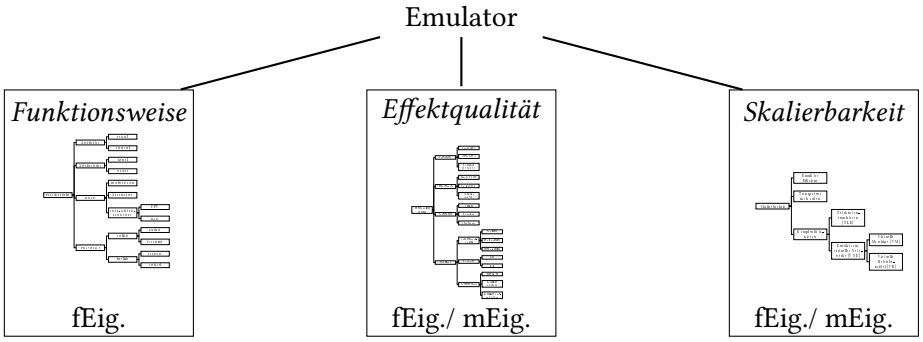


Abbildung 2.2: Hauptkategorien der entworfenen Taxonomie mit Angaben zu feststehen bzw. messbaren Eigenschaften.

Die *Funktionsweise* besteht allgemein aus feststehenden Eigenschaften und unterteilt sich in vier weitere Klassifikationsmerkmale, welche in Abschnitt 2.3 näher beschrieben werden.

Zur *Effektqualität* zählen die unterstützen Netzwerkeffekte sowie die Genauigkeit bei deren Umsetzung. Daher enthält dieses Klassifizierungsmerkmal neben den *feststehenden* auch *messbare* Eigenschaften. Die Angaben zur *Effektqualität* lassen sich zumeist aus den Veröffentlichungen zum jeweiligen Emulator entnehmen. Dennoch sind eigene Analysen bzgl. der Genauigkeit vorzuziehen, da die Betrachtungen in solchen Veröffentlichungen oft unter Idealbedingungen durchgeführt werden. Um den Grad der Genauigkeit zu einem bestimmten Netzwerkeffekt messen zu können, werden Metriken benötigt. In Abschnitt 2.4 werden daher Netzwerkmetriken häufig unterstützter Effekte vorgestellt.

Die *Skalierbarkeit* umfasst ebenfalls feste sowie messbare Merkmale Letztere beschreiben die mögliche Leistung und die dafür benötigten Ressourcen. Ein Beispiel dafür ist die CPU-Auslastung im Zusammenhang mit der maximal möglichen Paketrate bei der Emulation von verschieden vielen Knotenpunkten. Das mögliche Potenzial zur *Skalierbarkeit* ergibt sich aus den bereits erwähnten *feststehenden Eigenschaften* NLE und VNE. Aus diesem Grund werden dazu in Abschnitt 2.5 einige Grundprinzipien erläutert.

2.3 Funktionsweisen

Die Betrachtung der Funktionsweise eines Emulators hilft bei einer ersten Einschätzung bzgl. der Kosten, notwendigen Hardware, des Verwaltungsaufwands, möglichen Präzision etc. und gibt damit Aufschluss über dessen Eignung für eine bestimmte Testumgebung. Ist beispielsweise lediglich ein einzelner Rechner für den Testaufbau vorhanden, bleiben alle Emulatoren mit einem dezentralen Ansatz außen vor.

Die Funktionsweise eines Emulators wird in dieser Arbeit mit Hilfe der fünf Kategorien *Architektur*, *Arbeitsbereich*, *Zeitgeber*, *Bedarf* und *Pfadvorgabe* beschrieben. Abbildung 2.3 zeigt eine Übersicht der verwendeten Aufteilung.

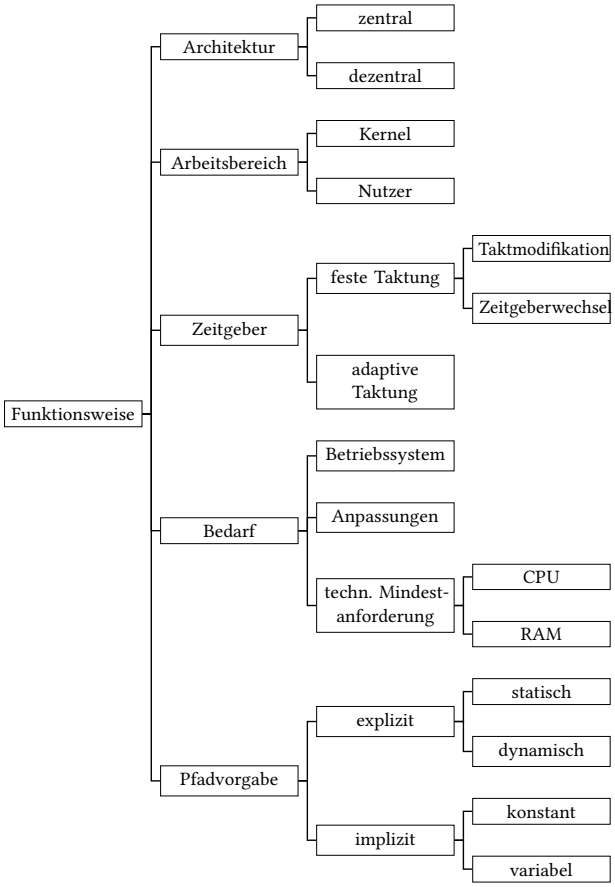


Abbildung 2.3: Kategorie *Funktionsweise* mit den fünf zugehörigen Unterkategorien.

2.3.1 Architektur

Diese Kategorie bezieht sich auf die real-existierende Hardware, welche zur Emulation eingesetzt wird. Ein Emulator kann dabei in einer eigens dafür konstruierten Hardware implementiert sein oder als Software vorliegen. Allgemein bestehen die Möglichkeiten den Emulator zentral auf einem einzelnen realen Netzwerkknoten *Kr* oder dezentral auf mehrere Rechnern verteilt zu installieren. In beiden Fällen kann durch eine weitere Virtualisierung eine komplexere Umgebung mit entsprechend virtuellen Knoten *Kv* aufgebaut werden als die zugrundeliegende Hardware ursprünglich ermöglicht hätte. So ist es beispielsweise denkbar eine Satellitenverbindung durch einen Emulator nachzuahmen, obwohl kein Satellit zur Verfügung steht.

Zentral Bei einem zentralen System befinden sich alle Emulationskomponenten auf einem einzelnen *Kr*. Es besteht die Möglichkeit, dass auch der Sender und/oder der Empfänger Teil dieses *Kr* ist/sind. Im allgemeinen Fall jedoch befindet sich der Emulator getrennt von Beiden auf einem gesonderten Rechner.

Die Zentralisierung hat den Vorteil, dass keine Uhrensynchronisation durchgeführt werden muss, da sich alle Emulatorelemente den gleichen Zeitgeber teilen. Dadurch sind zentrale Systeme

bei der Untersuchung von Paketverzögerungen am genauesten. Nachteile diese Architektur zeigen sich in der eingeschränkten Skalierbarkeit. Mit steigender Komplexität der Testumgebung steigt ebenfalls die benötigte Rechenleistung.

Dezentral Die dezentrale bzw. verteilte Lösung ist dagegen nicht an die Rechenleistung eines einzelnen Rechners gebunden und kann so mehr Knotenpunkte sowie Endpunkte virtualisieren. Für komplexere Testszenario besteht ein Emulator oft aus einem Verbund von Rechnern. Der Nachteil hierbei ist jedoch die nötige Synchronisation zwischen der verteilten Hardware, welche zusätzliche Verzögerungen verursacht. Lösungen, die von einer Synchronisation absehen und sich dadurch aus unabhängigen Rechnern zusammensetzen, sind in ihrer *Effektqualität* beschränkt.

Andere Nachteile bei dieser Architektur sind die höheren Kosten und der größere Verwaltungsaufwand.

2.3.2 Arbeitsbereich

Dieser Bereich stellt die Betriebssystemebene dar, in der ein Emulator Einfluss auf die Datenpakete ausübt. Der Arbeitsbereich kann sich daher in der Kernel-Ebene oder in der Nutzerebene befinden.

Wie präzise die Vorgaben für die Testumgebung umgesetzt werden, hängt überwiegend vom Manipulationsprozess und der Paketrate des Emulators ab. Dieser Prozess sollte so wenig zusätzliche Verarbeitungskosten (engl. *processing-overhead*) verursachen und damit Einfluss auf die eigentliche Paketabarbeitung haben wie möglich. Die Pakete werden idealerweise in Echtzeit manipuliert und weiterverarbeitet.

Kernel Arbeitet ein Emulator im Kernel-Bereich so ist der *processing-overhead* geringer als in der Nutzerebene, da Datenpakete zuerst vom Kernel verarbeitet und anschließend erst an die Nutzerebene weitergeleitet werden. Ein weiterer Vorteil ist die zeitnahe Vergabe von Zeitstempeln in diesem Bereich. Nachteilig wirkt sich die zusätzliche Anpassung des Kernel aus. So muss zumeist ein älterer Kernel gefunden, installiert, entsprechend angepasst und neu kompiliert werden.

Nutzer Eine Emulator im Nutzerbereich ist entsprechend einfacher zu installieren und kann dank eines aktuellen Kernels sogar auf die gleichen Zeitgeber zugreifen wie sonst bei älteren Systemen nur der Kernel. Dennoch bleibt der zusätzliche *processing-overhead* bestehen und die Paketrate fällt geringer aus als bei Emulatoren mit einem Arbeitsbereich im Kernel.

2.3.3 Zeitgeber

Die Leistung und Genauigkeit eines Emulators ist u. a. abhängig vom *Zeitgeber* sowie dessen *Taktrate*. In einem Rechnersystem stehen oft mehrere Zeitgeber zur Verfügung, wobei einer von diesen als Systemzeitgeber des Kernels fungiert. Mit jedem Takt werden je nach Rechnerarchitektur eine oder mehrere Instruktionen von der CPU durchgeführt. Eine Instruktion ist ein Befehl im Befehlsregister einer CPU und bildet zusammen mit weiteren Befehlen verschiedene Verarbeitungsprozesse im System. Auf einem System existieren mehrere Prozesse, welche für eine gewisse Anzahl an Takten die CPU in Anspruch nehmen dürfen. Prozesse können jedoch auch durch eine Unterbrechung (engl. *Interrupt*) den Anspruch kurzfristig verlieren und geben

diesen an den Unterbrechenden weiter. Ein Interrupt kann nur zusammen mit einem Takt erfolgen, da erst die aktuelle Instruktion beendet wird. Beispielsweise lösen ankommende bzw. ausgehende Pakete einen Interrupt aus[1]. Die Rate der Interrupts bzw. die *Taktrate* des Zeitgebers ist daher ausschlaggebend für die Verarbeitungsgeschwindigkeit und die Granularität bestimmter *Effektqualitäten*. Ist die Ein- bzw. Ausgangsfrequenz der Pakete höher als die Taktrate werden diese zwischengespeichert und gesammelt weiterverarbeitet. Bei einem zu geringen Zwischenspeicher werden Pakete jedoch verworfen. Um die Standardtaktrate des Systemzeitgebers zu erhöhen fordern Emulatoren entweder eine Anpassung der Frequenz oder geben dem System einen anderen Zeitgeber vor, dessen Taktrate höher ist. Ersteres ist mit einer Anpassung des Systems verbunden und nicht ohne eine Kompilierung des Kernels umzusetzen. Die andere Möglichkeit ist von der Installation her einfacher und ohne Anpassungen durchführbar. Jedoch sind nicht auf allen Systemen die gleichen Zeitgeber vorhanden. Wodurch ein entsprechender Emulator abhängig von einer bestimmten Hardware wird.

Eine zu hohe Taktrate birgt jedoch die Gefahr, dass das System eher verlangsamt wird, da der Wechsel zwischen den Prozessen ebenfalls mit einem Kontextwechsel verbunden ist. Die Zwischenergebnisse sowie Daten des unterbrochenen Prozesses werden in den Cache-Speicher gelegt und nach der Verarbeitung des Unterbrechenden Prozesses erneut angefordert. Die Skalierbarkeit des Emulators nimmt daher ab. Eine mögliche Lösung ist eine solch hohe Taktrate nur zeitlich begrenzt zu aktivieren bzw. *adaptive Zeitgeber* zu nutzen. Ein Beispiel dafür sind die *High-Resolution Timer* unter Linux, welche ereignisgesteuert die Taktung anpassen.

2.3.4 Bedarf

Der Bedarf umfasst die technischen Grundvoraussetzungen und die für die Emulation benötigten Ressourcen zur Generierung dieser Effekte. Dazu gehören das *Betriebssystem* und/oder die Kernelversion. Teilweise sind zusätzliche *Anpassungen* des Betriebssystems nötig, wie z. B. das Installieren eines Kernelmoduls oder einer weiteren Software. Diese Angaben findet man auf der Internetseite der Entwickler oder in den Handbüchern.

Die technischen *Mindestanforderung*, wie CPU oder RAM dagegen finden sich zumeist nur indirekt im Evaluierungsteil der Veröffentlichungen in den Angabe zum verwendeten Versuchsaufbau.

2.3.5 Pfadvorgabe

Die Definition einer Netzwerkverbindung kann *explizit* durch eine Liste von Werten vorgegeben oder *implizit* über die virtuelle Topologie beschrieben werden. Teile dieser Topologie können variabel gestaltet werden, um so z. B. Knotenausfälle und die damit verbundene Pfadumleitung nachzubilden. Die implizite Pfadvorgabe kann daher weiter in *konstante* oder *variable* Topologien eingeteilte werden. Die expliziten Vorgaben wiederum werden entweder *statisch* oder ereignisgesteuert bzw. *dynamisch* umgesetzt.

[1] Im Falle einer Interrupt-Verschmelzung können auch mehrere Pakete pro Interrupt verarbeitet werden. (vgl. [Büs13])

Explizit

Im expliziten Fall werden die Werte für bestimmte Netzwerkeffekte (vgl. Abs. 2.4) vor einem Versuchsdurchlauf an den Emulator übergeben.

Statisch Während eines Testdurchlaufs können diese Parameter nicht mehr angepasst werden. Was dazu führt, dass sich der Versuchsaufbau *statisch* verhält und dieser kaum mit einem realen Netzwerk vergleichbar ist. Der Vorteil für den Entwickler ist jedoch, dass dieser den Beobachtungsbereich für ein bestimmtes Szenario seinen Anforderungen entsprechend einschränken kann.

Dynamisch Bei ereignisgesteuerten Vorgaben wird der Manipulationsprozess des Emulators an eine bestimmte Begebenheit geknüpft. Das können beispielsweise eine gewisse Anzahl an empfangenen Paketen, eine Taktrate oder pseudozufällige Entscheidungen sein. Für größere Szenarien mit vielen Ereignissen wird oft die *SEmulation* benutzt.

Diese Art der Vorgabe ist für den generellen Funktionsnachweis eines Protokolls oder einer Anwendung besonders geeignet, da sich die Eigenschaften des emulierten Netzwerks *dynamisch* verhalten und trotzdem reproduzierbare Ergebnisse gemessen werden können. Während eines Testdurchlaufs kann z. B. für die Untersuchung eines Verkehrsleitsystems das Verhalten einer Netzwerkverbindung zu unterschiedlichen Tageszeiten nachgeahmt werden.

Implizit

Eine andere Möglichkeit bieten Netzwerkmodelle, welche teils aus realen oder auch komplett aus virtuellen Komponenten zusammengesetzt werden. Dadurch lassen sich für den Versuchsaufbau Netzwerktopologien nachbilden, die der späteren Zielplattform entsprechen. Die Bausteine bilden beispielsweise Router mit definierten Eigenschaften nach und können zentral oder dezentral platziert sein. Bei einer zentralen Installation werden zumeist die virtuellen Endpunkte auf einem einzelnen Rechner zusammengefasst und die emulierten Netzwerkknoten werden verteilt. Versendet werden reale Datenpakete, welche durch die Eigenschaften der realen oder virtuellen Komponenten beeinflusst werden. Das globale Netzwerkverhalten und die Einflüsse innerhalb der emulierten Topologie setzen sich somit, wie auch in der Realität, aus dem Verhalten der einzelnen Komponenten zusammen.

Konstant Bei diesen Topologien bleiben die Verbindungspfade aus Sicht der Endsysteme unverändert. Dadurch lassen sich Experimente durchführen, die den Fokus auf einen Pfad mit einer bestimmten Verkettung von Komponenten legen. Beispielsweise bei der Nachbildung eines Feldbusses in der Gebäudeautomatisierung. Vorteilhaft ist die im Gegensatz zur variablen Pfadvorgabe höhere *Skalierbarkeit* (vgl. Abschnitt 2.5), da keine redundanten Pfadverbindungen vorgehalten werden müssen. Es kann sich jedoch gleichzeitig als Nachteil erweisen, wenn unterschiedliche Verkettungen überprüft werden sollen und somit für jede Variante ein neues Experiment gestartet werden muss.

Variabel Eine veränderliche Topologie dagegen kann die verschiedenen Varianten und darüber hinaus bestimmte Ereignisse nachbilden. Dazu zählen zum Beispiel Verbindungsabbrüche

oder das im Mobilfunk relevante *Handover* beim Wechseln der Funkzelle. Dadurch gestattet diese Art der Pfadvorgabe die beste Nachbildung von realen Netzwerken.

Implizite Pfadvorgaben entsprechen am ehesten realen Umgebungen, können jedoch kontrolliert Ausnahmesituationen auslösen. Im Vergleich zu expliziten Vorgaben sind diese abstrakt gehalten und erlauben so eine einfachere Umsetzung von komplexen Netzwerkstrukturen. Bei den expliziten Vorgaben werden dagegen im Vorfeld Annahmen über das Pfadverhalten getroffen bzw. es wird auf eine realitätsnahe Testumgebung verzichtet und nur bestimmte Effekte stehen im Fokus der Untersuchung.

2.4 Effektqualität/Netzwerkmetriken

In einer vorangegangenen Arbeit[Büs13] wurde ein Messtool zur Analyse von Ende-zu-Ende Verbindungen entwickelt. Die folgenden Erläuterungen und Abbildungen zu den Netzwerkmetriken stammen in abgewandelter Form aus dieser Arbeit.

Die Werte von Netzwerkmetriken geben Auskunft über die Eigenschaft eines Verbindungspfades und repräsentieren die Einflüsse auf die betrachteten Datenpakete. Ein Verbindungspfad P besteht dabei aus $i = 1$ bis H Teilabschnitten (engl. *hops*) (vgl. Abb. 2.4). Bei einer Emulation

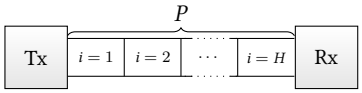


Abbildung 2.4: Pfad P mit den Teilabschnitten von $i = 1$ bis $i = H$ (Quelle: [Büs13])

werden diese Werte vorgegeben, um so die Eigenschaft eines Testpfades weitestgehend zu kontrollieren. Der Grad der Umsetzung dieser Vorgaben durch einen Emulator ist Gegenstand der Genauigkeitsanalyse.

Zu den in der Genauigkeitsanalyse betrachteten Netzwerkmetriken zählen neben der Datenrate auch Metriken, welche speziell in paketbasierten Netzwerken auftreten. Dazu gehören Paketverlustraten und -dopplungsraten sowie Änderungen in der Paketreihenfolge. Zudem ist die Paketverzögerung und deren Variation für die Betrachtung der Genauigkeit eines Emulators entscheidend. Eine Übersicht der Metriken ist in Abbildung 2.5 zu finden. Jede Einzelne davon wird im Folgenden definiert.

2.4.1 Datenraten

Die meisten Emulatoren bieten eine Einstellungsmöglichkeit für die maximal erreichbare Übertragungsrate an. Diese kann Werte kleiner oder gleich der physischen Maximalrate annehmen. Allgemein ist die maximale Datenübertragungsrate (engl. Capacity C) einer Netzwerkverbindung zeitunabhängig und wird durch die physische Kanalkapazität beschränkt. Auf einen Verbindungspfad bezogen ist C die minimale Einzeldatenrate C_i der Teilabschnitte (Abb. 2.6). Als Gleichung, wobei C_0 der Senderate entspricht (vgl. [Pra+03]):

$$C = \min_{i=0,\ldots,H} C_i \tag{2.1}$$

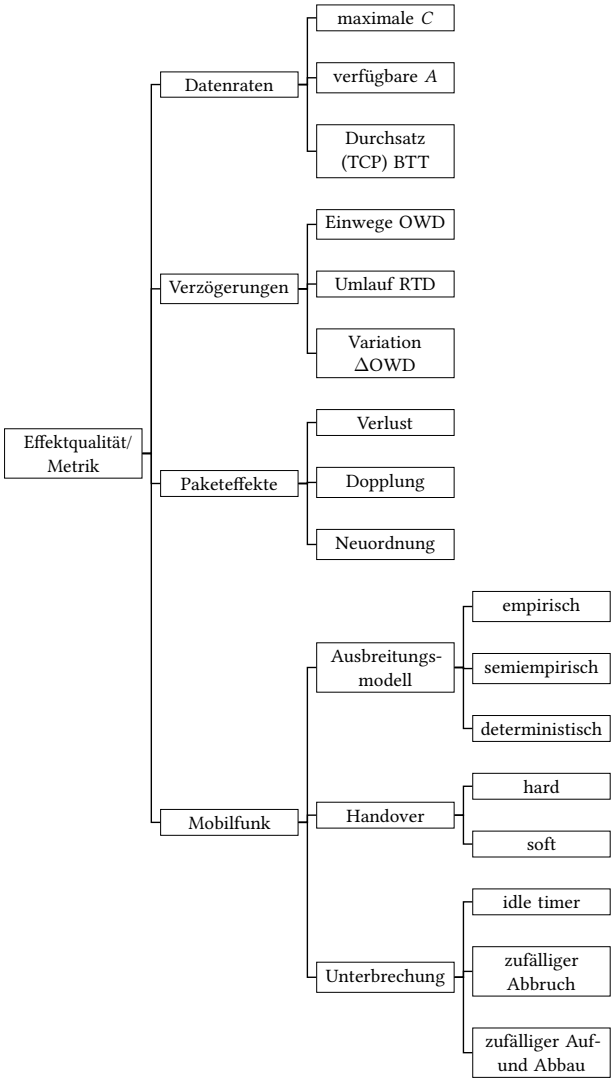


Abbildung 2.5: Kategorie *Effektqualität* mit den verwendeten Metriken.

Im einfachsten Fall beschränkt ein Emulator diese physische C auf den geforderten Wert durch zusätzliche Paketverzögerungen.

Einige Emulatoren sind in der Lage Hintergrundverkehr (engl. *Cross Traffic* (CT)) nachzuahmen, welcher indirekt über die verfügbare Datenrate (engl. Available Bandwidth A) gemessen werden kann. Diese ist zeitabhängig und wird direkt vom CT beeinflusst. Der Wert von CT entspricht der Auslastung (engl. *utilization* u_i) des jeweiligen Teilabschnittes und beschränkt die zugehörige Datenrate A_i. Die Werte von u_i liegen im Bereich $0 \leq u_i \leq 1$. In einem Pfad muss nicht zwingend der Teilabschnitt mit der minimalen C_i auch die geringste verfügbare Datenrate beinhalten. Die Formel für A bzgl. der Gesamtverbindung P lautet, wobei u_i und A sich im gleichen Zeitintervall befinden(vgl. [Pra+03]):

$$A = \min_{i=0,\ldots,H} (C_i(1 - u_i)) \tag{2.2}$$

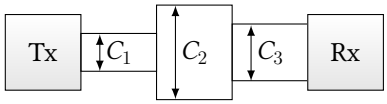


Abbildung 2.6: P mit $H = 3$ und $C = C_1$ (Quelle: [Büs13])

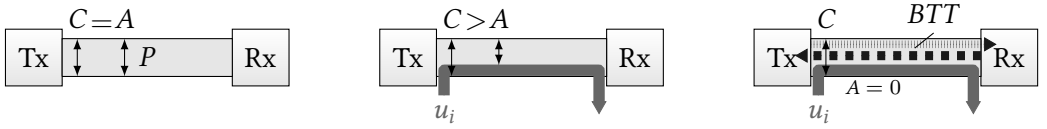
Ein Umstellen der Gleichung 2.2 nach u_i ergibt:

$$\max_{i=0,\ldots,H}(u_i) = 1 - \frac{A}{C} \tag{2.3}$$

Zu Beachten ist, dass sich beide Gleichungen auf den gesamten Verbindungspfad beziehen. Die Gesamtauslastung u ist daher gleich der maximalen Einzelauslastung u_i.

Um auch Aussagen über die Genauigkeit der emulierten Datenrate bei der Verwendung von TCP-Paketen treffen zu können, wird auch die Durchsatzrate betrachtet. Anderes als die maximale und die verfügbare Datenrate bezieht sich diese Größe auf die Nutzdatenrate und wird von den Protokoll-spezifischen Mechanismen beeinflusst. Der maximal erreichbare Durchsatz einer TCP-Verbindung wird als Massendurchsatzrate (engl. *Bulk TCP Throughput* (BTT)) bezeichnet.

Abbildung 2.7 verdeutlicht den unterschiedlichen Kontext der genannten Datenraten. In 7.1a



(a) max. Datenrate entspricht der verfügbaren Datenrate

(b) zusätzliche Last verringert verfügbare Datenrate

(c) Unterschied zwischen C,A gegenüber BTT

Abbildung 2.7: Definition der verschiedenen Raten. (Quelle: [Büs13])

unterscheiden sich die Werte von A und C nicht, da keine weitere Last u_i vorhanden ist. Der Wert von A sinkt sobald eine zusätzliche Last auf das Übertragungsmedium wirkt (siehe 7.1b), während der Wert von C laut Definition unverändert bleibt. In Abbildung 7.1c wird der Unterschied zur BTT verdeutlicht. Eine TCP-Verbindung besteht, wegen gegenseitiger Steuerinformationen, aus zwei Richtungen, wobei der Durchsatz während einer Verbindung verschieden oft angepasst wird. Beispielsweise wird durch eine langsame Start-Senderate (engl. *Slow-Start*, aus [All+09, S.4]) anfangs nicht die verfügbare Datenrate genutzt. Wird der maximale Durchsatz erreicht, ist A gleich Null, jedoch stimmt der Wert der BTT nicht mit dem Wert von A vor der TCP-Verbindung überein.

2.4.2 Verzögerung

Die folgenden Metriken beziehen sich auf die Paketlaufzeiten aus Sicht der Anwendung. Die zusätzlichen Latenzen, die durch die Emulation entstehen, werden beispielsweise in Abb. 3.6 gezeigt und gehören mit zur Analyse.

Die Verzögerung (engl. *Delay*) ist zeitlich beschränkt und ist die Differenz zwischen dem Sendezeitpunkt t^{Tx} und Empfangszeitpunkt t^{Rx} eines Paketes. Je nach Betrachtungsweise befinden

sich dabei Sender T_x und Empfänger R_x auf unterschiedlichen oder aber auf ein und derselben Maschine. Im ersten Fall wird die Verzögerung in eine Richtung (engl. *One Way Delay* (OWD), aus [AKZ99a]) der Verbindung gemessen. Im anderen Fall wird die Umlaufverzögerung (engl. *Round Trip Delay* (RTD), aus [AKZ99b]) betrachtet. Wichtig ist, dass im ersten Fall eine Synchronisation der Zeitgeber auf beiden Seiten nötig ist. Die Verzögerung eines Paketes setzt sich unter anderem aus den Einzelverzögerungen bei der Übertragung D_T^j, Ausbreitung D_P^j und in der Warteschlange D_Q^j zusammen[2].

Die Darstellung in Bild 2.8 veranschaulicht die Zusammensetzung der Einzelverzögerung für eine Richtung.

Formel 2.4 zeigt die Berechnung für die jeweilige Verzögerung des j-ten Paketes:

$$t^{Rx} - t^{Tx} = \begin{cases} D_T^j + D_P^j + D_Q^j, & \text{OWD} \\ D_T^j + D_P^j + D_Q^j + D_{T'}^j + D_{P'}^j + D_{Q'}^j & \text{RTD} \end{cases} \tag{2.4}$$

Das Hochkomma bei den Einzelverzögerungen der RTD zeigt an, dass die Einzelverzögerungen bei der Hin- und Rückrichtung durchaus unterschiedliche Werte annehmen können. Einige Emulatoren sind in der Lage ein- und ausgehende Pakete unterschiedlich zu behandeln, um so asynchrone Verbindungen nachzubilden.

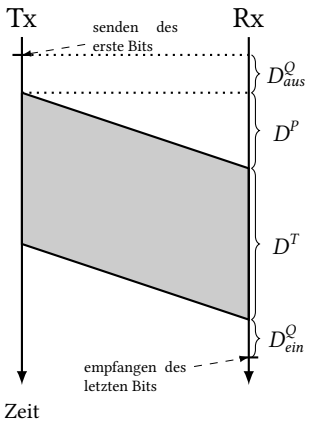


Abbildung 2.8: Schema der Verzögerungsphasen bei OWD. (Quelle: [Büs13])

Zufällige Schwankungen der Paketlaufzeiten verursachen unterschiedliche Verzögerungen. Speziell für die OWD hat sich der Begriff *Jitter* trotz Mehrdeutigkeit für die Variation der Verzögerungen zwischen aufeinanderfolgenden Paketen etabliert. Eine Veranschaulichung dazu ist in Abbildung 2.9 zu sehen. Als Gleichung, wobei ΔOWD^j dem *Jitter* des j-ten Paket entspricht:

$$\Delta\text{OWD}^j = \text{OWD}^j - \text{OWD}^{j-1} = (D_T^j - D_T^{j-1}) + (D_P^j - D_P^{j-1}) + (D_Q^j - D_Q^{j-1}) \tag{2.5}$$

Geht man davon aus, dass die Pakete mit der selben Größe über den gleichen Pfad versendet werden, so vereinfacht sich Gleichung (2.5) zu:

$$\Delta\text{OWD}^j = D_Q^j - D_Q^{j-1} \tag{2.6}$$

[2]Je nach Anwendung können diese Verzögerungen noch weiter aufgeteilt werden.

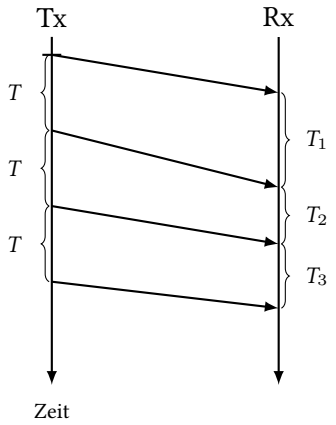


Abbildung 2.9: Verzögerungsvariationen. (Quelle: [Büs13])

Die Ausbreitungs- und Transferverzögerungen verhalten sich unter dieser Annahme gleichbleibend. Somit ist D_Q^j die ausschlaggebende Größe für *Jitter*. Für die Genauigkeitsanalyse ist die erreichbare Präzision bzw. die umsetzbare Granularität für die ΔOWD^j ein ausschlaggebender Punkt.

2.4.3 Paketeffekte

Die Effekte *Paketverlust, -dopplung sowie -neuordnung* treten ebenso wie *Bitfehler* in realen Testumgebung selten und unkontrolliert auf. Ein Hauptgrund für den Einsatz von Emulatoren ist Verbindungspfade dahingehend gezielt manipulieren zu können, um so das Programmverhalten auch in Ausnahmesituationen untersuchen zu können.

Auch hier bestehen die Möglichkeiten eine einzelne Richtung oder den Hin- und Rückweg zu betrachten. Jedoch unterscheiden sich die Berechnungen für die beiden Möglichkeiten nicht voneinander. Daher werden die folgenden Raten allgemein betrachtet.

Ein Vergleich zwischen der Menge der empfangenen Pakete M_{Rx} und der Menge der gesendeten Pakete M_{Tx} gibt Auskunft über die jeweilige Rate.

Für die Paketverlustrate (engl. *Packet Loss Rate* (PLR)) wird das Verhältnis aus den beiden genannten Mengen gebildet:

$$\text{PLR} = 1 - \frac{M_{Rx}}{M_{Tx}} \tag{2.7}$$

In der Praxis wird dieses Verhältnis nicht erst am Ende einer Datenübertragung berechnet, sondern es werden Zeitfenster und aufsteigende Paketnummern verwendet, um so frühzeitig Verluste zu erkennen. Ein Paket gilt als verloren, wenn Pakete mit einer höheren Nummerierung am Empfänger auftreffen und das Zeitfenster geschlossen ist. Zeitfenstern können jedoch zu einem weiteren Effekt führen. Ist das Zeitfenster zu klein gewählt und wird ein verlässliches Protokoll wie z. B. TCP verwendet, so entsteht eine Wiederholungsrate (engl. *Retransmission Rate* (RR)), welche größer als die tatsächliche PLR ist. Dadurch kommt es zu Paketdopplungen. Entsprechend lässt sich daher auch eine Paketdopplungsrate (engl. *Packet Duplication Rate* (PDR))

angeben, die das Verhältnis aus der Menge der empfangenen doppelten Pakete D_{Rx} und der Menge der versendeten Pakete darstellt:

$$\mathrm{PDR} = \frac{D_{Rx}}{M_{Tx}} \tag{2.8}$$

Der dritte Effekt, die Paketneuordnung, wird bei Netzwerkmessungen häufig anstelle der PDR betrachtet bzw. es wird nicht explizit zwischen doppelten und neu geordneten Paketen unterschieden. Entsprechende Analysetools zählen daher alle doppelten Pakete zu den neu geordneten.

Da jedoch beide Raten getrennt voneinander emuliert werden können, ist eine Unterscheidung zwischen Paketdopplungen und Paketneuordnungen für die Genauigkeitsanalyse erforderlich.

Pakete die außerhalb einer definierten Reihenfolge beim Empfänger auftreffen gelten als ungeordnet und müssen neu eingereiht werden. Der englische Begriff dafür lautet *reordering*, daher wird in dieser Arbeit die Bezeichnung *Neuordnung* verwendet.

Der Wert der Neuordnungsrate (engl. *Paket Reordering Rate* (PRR)) gibt das Verhältnis zwischen der Menge der gesendeten und der Menge empfangenen ungeordneten Pakete N_{Rx} innerhalb eines Paketstromes an:

$$\mathrm{PRR} = \frac{N_{Rx}}{M_{Tx}} \tag{2.9}$$

Zumeist werden aufsteigende Sequenznummern für Ermittlung der ungeordneten Pakete verwendet. Dabei ist die Betrachtungsweise wichtig, ab wann ein Paket als außerhalb der Reihenfolge gilt. In dieser Arbeit gelten alle verspäteten Pakete als ungeordnet, d. h. diese Pakete treffen erst nach einem ursprünglichen Folge-Paket ein.

Der letzte Paketeffekt, der auf veränderte Bits während der Übertragung beruht, wird in der Praxis häufig dem Paketverlust gleichgesetzt. Im einfachsten Fall nimmt ein einzelnes Bit den gegenteiligen Wert an und wird vom Empfänger als fehlerhaft erkannt. Mehrere Bitfehler innerhalb eines Datenpaketes können durch entsprechende Kanalkodierungskodes bis zu einem gewissen Maß korrigiert werden. Darüber hinaus jedoch, ist das Paket für den Empfänger ohne Bedeutung und wird verworfen. Die Häufigkeit mit der Bitfehler auftreten, wird zumeist als Bitfehlerrate[3] (engl. *Bit Error Rate* (BER)) bezeichnet. Emulatoren, welche die Nachbildung von Bitfehlern gestatten, nehmen diese Rate oft als Verhältnis entgegen. Beispielsweise besagt eine BER von 10^{-3}, dass jedes tausendste Bit fehlerhaft ist.

Für kabelgebundene Verbindung spielt die BER eine geringe Rolle. Ganz anders verhält es sich jedoch in kabellosen Netzwerken.

2.4.4 Mobilfunk

Für die Nachbildung von Verbindungen im Mobilfunkbereich werden Emulatoren benötigt, welche funkspezifische Merkmale unterstützen. Dazu zählen allgemein *Ausbreitungsmodelle*, *Bewegungen* und speziell *Handover*-Effekte sowie *Unterbrechungen* der Verbindungen.

Ausbreitungsmodell Funkbasierte Verbindungen werden durch ihre Empfangsqualität charakterisiert. Diese ist wiederum das Resultat aus komplexen Umgebungseigenschaften und der

[3] Diese Bezeichnung wurde dem Englisch entnommen. Die korrekte deutsche Bez. lautet Bitfehlerverhältnis.

eingesetzten Technik. Zu ersteren zählen z. B. Reflexionen und Absorptionen. Bei der Funktechnik spielen u. a. die Antennen und die Sendeleistung eine Rolle.

Modelle, welche die Empfangsqualität anhand der Frequenz, elektromagnetischer Störeinflüsse, örtliche Gegebenheiten, Funktechnik und des Abstandes vom Sender für einen begrenzten Bereich berechnen, heißen Ausbreitungsmodelle. Der Grad der Komplexität der genannten Einflüsse bestimmt den Grad der Genauigkeit des Models. Die Werte dieser Einflüsse beruhen auf empirischen, semiempirischen und deterministischen Daten bzw. Berechnungen. Entsprechend lassen sich Ausbreitungsmodelle klassifizieren. Wobei der Grad der Genauigkeit:

- *empirischer* Modelle auf ähnlichen Messungen sowie Erfahrungswerten beruht und daher gering ist. Beispiel: Okumura-Hata Modell
- *semiempirischer* Modelle auf Berechnungen mit Vereinfachungen und Näherungen beruht und somit mittelmäßig ist. Beispiel: COST 231 LOS Modell
- *deterministischer* Modelle auf exakten Berechnungen beruht und dementsprechend hoch ist. Beispiel: Ikegami Modell

Bewegungen Mit Hilfe der Ausbreitungsmodelle ist es möglich innerhalb eines begrenzten Bereiches einem Ort eine bestimmte Empfangsqualität zuzuordnen. Ändert sich der Ort, ändert sich zumeist die Empfangsqualität. Emulatoren die Bewegungen unterstützen, arbeiten oft mit Simulatoren zusammen. Mit diesen werden im Vorfeld einer Emulation die Bewegungen simuliert und mit Hilfe eines Ausbreitungsmodells die zugehörigen Empfangsqualitäten berechnet. Anschließend werden die Ergebnisse vom Emulator zeitlich versetzt mittels Verzögerungen, Datenraten und weiteren Paketeffekten umgesetzt.

Handover Bewegungen rufen weitere Effekten hervor. Speziell für die Emulation zellular aufgebauter Funknetze ist die Nachbildung von Zellwechseln naheliegend. Aus Sicht eines Mobilfunkgerätes findet eine Übergabe (engl. *Handover*) von einer Funkzelle in die andere statt. Je nach Implementation kann es dabei zu einem kurzzeitigen Verbindungsabbruch (*Hard-Handover*) oder einer Parallelverbindung (*Soft-Handover*) kommen. Bei letzteren verbindet sich das Mobilfunkgerät zuerst mit der neue Zelle und bricht dann die Verbindung zur alten Zelle ab. Beim *Hard-Handover* dagegen beendet das Gerät zuerst die Verbindung zur alten Zelle und verbindet sich danach mit der Neuen.

Unterbrechungen Einige Emulatoren gestatten eine kontrollierte Unterbrechung einer Verbindung, um das Verhalten von Protokollen und verteilten Anwendungen bei Verbindungsabbrüchen untersuchen zu können. So kann z. B. das zuvor beschriebene *Hard-Handover* nachgebildet werden.

Nach [NK11] können Unterbrechungen weiter unterschieden werden in:

- Abbruch nach einer bestimmten Leerlaufzeit (engl. *Idle timer disconnection*)
- Zufälliger Abbruch
- Zufälliger Auf- und Abbau der Verbindung

2.5 Skalierbarkeit

Testumgebungen benötigen oft die Emulation einer hohen Anzahl an Endsystemen und Netzwerkknoten.

Emulatoren nutzen dazu allgemein die Tatsache aus, dass die zu untersuchenden Protokolle oder Anwendungen zumeist nur einen geringen Anteil der Betriebsmittel eines Rechners benötigen. Es verschafft diesen die Möglichkeit die vorhandenen Mittel aufzuteilen und auf einem einzelnen *Kr* mehrere Emulationseinheiten zu starten.

Dieser Umstand führt zum Begriff der *Skalierbarkeit* eines Emulators, welche aus der Sicht des Testers Auskunft über die mögliche Anzahl an Endsystemen und Netzwerkknoten im geplanten Versuchsaufbau gibt. Die Emulationseinheiten dürfen dabei jedoch nur so viele Ressourcen beanspruchen, dass dadurch die eigentliche Anwendung nicht beeinflusst wird. Abbildung 2.10 zeigt die Faktoren, welche für die Skalierbarkeit eines Emulators eine Rolle spielen.

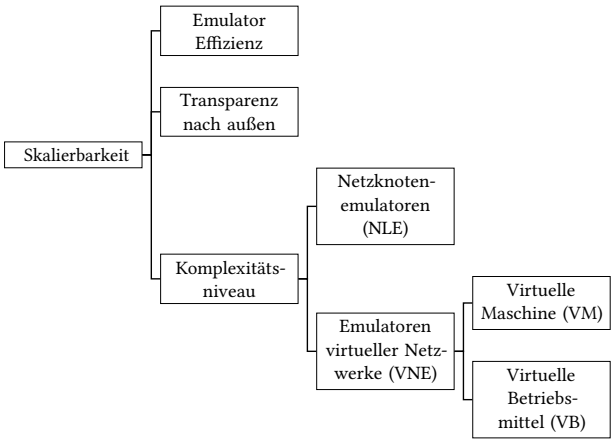


Abbildung 2.10: Kategorie *Skalierbarkeit* und deren Einflussfaktoren.

Aus Sicht der Testanwendung muss der Emulationsprozess *transparent* sein. Somit beschränkt die Transparenz die *Skalierbarkeit*.

Um möglichst viele Emulationseinheiten für eine begrenzte Anzahl an Betriebsmitteln generieren zu können, muss ein Emulator dazu möglichst *effizient* arbeiten. Daher bestimmt der Grad der Effizienz den Grad der Skalierbarkeit.

In Folge dessen wurden verschiedenste Methode entwickelt, um eine möglichst hohe Effizienz bei der Emulation zu erreichen und trotzdem die Transparenz zu wahren.

Deren Umsetzungen können dabei in die bereits erwähnten Netzknotenemulatoren NLE und Virtuelle-Netzwerke Emulatoren VNE eingeteilt werden. Gleichzeitig zeigt diese Einteilung das mögliche Komplexitätsniveau und damit auch die Skalierbarkeit an.

Netzknotenemulationen NLE beschränken sich auf die Nachahmung von einzelnen oder mehreren Knotenpunkten. Für die Endsysteme werden zusätzliche Rechner in der Versuchsumgebung benötigt. Mehrere Knotenpunkte lassen sich durch eine Aufteilung der Paketverarbeitung in einzelne Abarbeitungsstationen erreichen. Ein eingehendes Paket durchläuft je nach Vorgabe eine bestimmte Anzahl dieser Stationen. Eine Station entspricht

dabei einem Pfadabschnitt mit definierten Eigenschaften wie z. B. einer Verzögerung. Dadurch lassen sich Verbindungspfade durch eine Netzwerktopologie erstellen.

Emulationen virtueller Netzwerke VNE können darüber hinaus Endsysteme nachbilden. Neben der Partitionierung verwenden diese Emulatoren zusätzliche Methoden der Virtualisierung und Simulation. Letztere werden im Teilabschnitt zur *SEmulation* zusammengefasst.

Die Methoden zur Virtualisierung lassen sich weiter aufteilen in die Virtualisierung ganzer Rechner (engl. *Virtual Machine* (VM)) oder einzelner Betriebsmittel (VB) wie dem Protokollstapel. Der Nachteil von VMs ist deren Redundanz. So beinhaltet beispielsweise jede VM ein eigenes Betriebssystem, dass zur Funktion einer Mindestanforderung an Betriebsmitteln bedarf, welche jedoch über den Anforderungen der eigentlichen Emulation liegen. Die Skalierbarkeit ist entsprechend geringer als bei VB.

Bei VB dagegen werden lediglich die benötigten Ressourcen virtualisiert und aufgeteilt. Ein Beispiel dafür ist das unter FreeBSD verfügbare Jails [13d], welches eine Partitionierung in mehrere kleine Instanzen des System erlaubt. Ein Emulator kann so mehrere *Kv* unter einem Betriebssystem erstellen.

Dennoch benötigt auch in diesem Fall die Virtualisierung selbst einen Teil der Ressourcen.

Die Skalierbarkeit eines Emulators bezieht sich im Fall von NLE auf die Anzahl der Pfadabschnitte (*hops*) unter der Betrachtung der maximal möglichen Paketrate. Während bei VNE zusätzlich die Anzahl der Endpunkte betrachtet werden kann.

2.6 Zusammenfassung

In diesem Kapitel wird zunächst der Begriff der Emulation klar von dem der Simulation und SEmulation abgegrenzt. Anschließend werden traditionelle Klassifikationen für Emulatoren erweitert, um grundlegende Unterscheidungsmerkmale zu erläutern. Diese Merkmale werden dabei in *feststehende* und *messbare* unterschieden. Die feststehenden Eigenschaften werden einzeln aufgeführt sowie diesbezügliche Vor- und Nachteile aufgezeigt. Für die messbaren Eigenschaften werden entsprechende Metriken vorgestellt, welche in einem eigenen Messkonzept Anwendung finden.

Gleichzeitig lässt sich aus den Erklärungen und Angaben eine eigene angepasste Taxonomie erhalten. Mit Hilfe der vorgestellten Taxonomie lassen sich im Vorfeld Einsatzgebiete und Anwendungsmöglichkeiten für Emulatoren bestimmen. Die Aussagen zu einzelnen *feststehenden Merkmalen* können für bestimmte Aspekte in einen Zusammenhang gebracht werden, um so ein Bewertungsschema für Emulatoren zu erstellen. Ein Beispiel dafür ist in Tabelle 2.1 zu sehen. Diese Tabelle fasst einige der genannten Vorzüge und Einschränkungen vieler *feststehenden Eigenschaften* zusammen, um dadurch z. B. Aussagen über die Verarbeitungsrate, den Installationsaufwand, die Effektgranularität etc. treffen zu können. So ist beispielsweise der Installationsaufwand bei Emulatoren mit einer zentralen *Architektur*, welche anstatt einer Kernel-Anpassung einfach einen anderen Zeitgeber verwenden und statische Pfadvorgaben, ohne die Notwendigkeit einer Topologievorgabe unterstützen, im positiven Sinne gering. Für den Aspekt der Realitätsnähe sind Emulatoren mit impliziter Pfadvorgabe und der Möglichkeit virtuelle Netzwerke zu erstellen besonders geeignet. Allerdings leidet die Skalierbarkeit und Verarbeitungsrate

unter dem Einsatz von VMs und implizierten variablen Pfaden, da in beiden Fällen Redundanzen entstehen. Das vorgestellte Beispiel eines Bewertungsschemas ist diskussionswürdig und beinhaltet einige verallgemeinerte Einschätzungen, welche im Einzelnen vom jeweiligen Emulatoren abhängen und mit Leistungs- und Genauigkeitsanalysen geprüft werden müssen.

Im folgenden Kapitel werden die Funktionsprinzipien einer breit gefächerten Auswahl von Emulatoren vorgestellt und gleichzeitig Evaluationen sowie Studien zu diesen erläutert. Anhand dieser Prinzipien lassen sich die Emulatoren in die vorgestellte Taxonomie einordnen, um z. B. einen Grundstock für die Einschätzungen zu erhalten. Außerdem lässt sich so ein eigenes Messkonzept ableiten, mit dessen Hilfe u. a. Werte gefunden werden können, welche die getroffenen Einschätzungen entweder untermauern oder eine Berichtigung dieser erfordern.

				Synchronisationsaufwand	Skalierbarkeit	Verarbeitungsrate	Effektgranularität	Installationsaufwand	Administration	Topologiekomplexität	Realitätsnähe
Funktionsweise	Architektur	zentral		+	−	+		+	+	−	
		dezentral		−	+	*o*		−	−	+	
	Zeitgeber	feste Taktung	Modifikation		−	*o*	*o*	−			
			Wechsel		+	+	+	+			
		adaptive Taktung			*o*	+	+	+			
	Arbeitsbereich	Kernel			+	+	+	*o*	−	+	
		User			−	−	−	+	+	−	
	Pfadvorgabe	explizit	statisch	+	+	+		+	+	*o*	−
			dynamisch	*o*	+	*o*		+	+	*o*	*o*
		implizit	konstant	*o*	*o*	−		*o*	−	+	+
			variabel	−	−	−		−	−	+	+
Skalierung	Netzknotenemulation			+	+	+		+	+	*o*	*o*
	Emulation virtueller Netzwerke		VM	−	−	−		−	*o*	+	+
			VB	+	*o*	*o*		*o*	+	+	+

Tabelle 2.1: Beispiel für ein Bewertungsschema einiger ausgewählter *feststehenden Eigenschaften* der Klassen *Funktionsweise* und *Skalierbarkeit* anhand der genannten Vor- und Nachteile. (Bedeutung: „+“-positiver, „−“-negativer, „*o*“-mittelmäßiger und „ “-kein Zusammenhang)

3 Verwandte Arbeiten

Im Hinblick auf die Anforderungsanalyse und die Schaffung eines eigenen Messkonzepts, werden in diesem Kapitel verwandte Arbeiten betrachtet. Neben einigen wenigen neutralen Vergleichsstudien, sind es vor allem die Entwickler des jeweiligen Emulators, die Betrachtungen zur Genauigkeit und Leistung anstellen. Teilweise vergleichen diese ihr Produkt mit anderen Emulatoren, um so einen besonderen Gütegrad bestimmter Merkmale hervorzuheben. Dadurch wiederum ergeben sich Hinweise für ein eigenes Messkonzept.

Somit wird in diesem Kapitel zunächst die Auswahl der Emulatoren erläutert und anschließend auf deren Publikationen eingegangen. Abschließend werden neutrale Vergleichsstudien vorgestellt.

3.1 Auswahl der Emulatoren

Die Auswahl der Emulatoren orientiert sich in erster Linie an deren Verfügbarkeit und Bekanntheitsgrad. Dadurch ist es möglich eigene Untersuchungen durchzuführen und es kann davon ausgegangen werden, dass zu diesen Emulatoren bereits Vergleichsuntersuchungen existieren, welche wiederum Anhaltspunkte für ein eigenes Messkonzept liefern.

Weiterhin wird, mit Blick auf die im Kapitel 2 vorgestellte Taxonomie versucht, eine möglichst breitgefächerte Auswahl zu treffen.

Dem Institut steht ein Hardware-Emulator „Linktropy 7500 Pro“ des Herstellers Apposite [13a] zur Verfügung, welcher bereits in einer vorangegangenen Arbeit [Büs13] als Referenzpunkt für Genauigkeitsuntersuchungen diente. Eine Publikation mit Analysen zu dessen Genauigkeit oder Leistung ist nicht auffindbar. Allerdings werden in der zugehörigen Produktdokumentation Angaben zu Abweichungen und zur Verarbeitungsgeschwindigkeit gemacht. Mit diesen Werten lassen sich wiederum in eigenen Untersuchungen Vergleiche anstellen. Daher wird dieser Hardware-Emulator trotz fehlender Analysen mit betrachtet.

Alle folgenden Emulatoren sind Software-basiert und können als Anwendung, Kernelpatch oder VM aus dem Internet heruntergeladen werden. Über eine Trendanalyse von Internetsuchanfragen lässt sich eine erste Auswahl treffen und der Bekanntheitsgrad sowie der ungefähre Zeitpunkt der Veröffentlichung ablesen. In Abbildung 3.1 ist eine solche Analyse für Suchanfragen an den Suchmaschinenbetreiber Google zu sehen. Der Zeitbereich beginnt im Januar 2004, da erst ab diesem Zeitpunkt die Daten zur Analyse veröffentlicht wurden.

Zu sehen ist, das Dummynet [13w] einer der ersten und bekanntesten Emulatoren ist. In der jüngsten Dokumentation [CR10] dazu finden sich Aussagen zu weiteren Emulatoren wie z. B. KauNet[13j], ModelNet[13n] oder Emulab[13c], welche allesamt auf Dummynet aufbauen. Eine andere Analyse, welche mit Hilfe der für wissenschaftliche Arbeiten spezialisierte Suchmaschine Google Scholar[13e] durchgeführt wurde, ist in Abb. 3.2 zu sehen. Diese zeigt die Häufigkeit an, mit der die erste wissenschaftliche Veröffentlichung zum jeweiligen Emulator zitiert wurde.

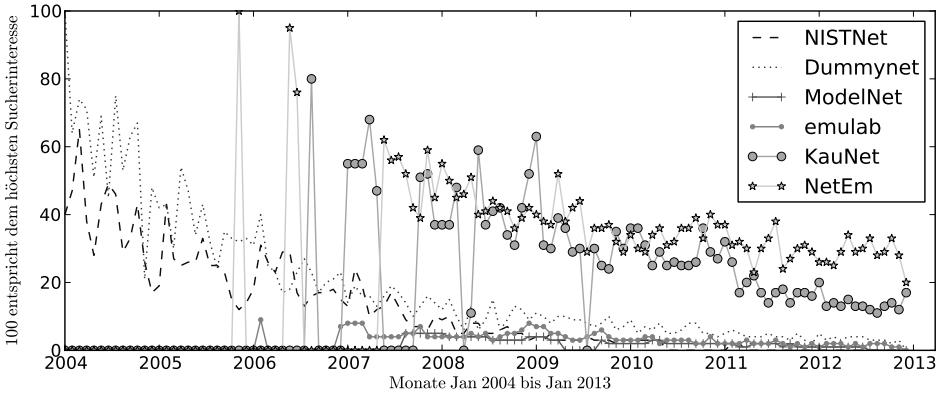


Abbildung 3.1: Analyse der Suchanfragen zwischen 2004 und 2013 für bekannte Emulatoren. (Quelle: [13f])

Dummynet zeichnet sich dabei deutlich von den anderen Emulatoren ab, was jedoch auch darauf zurück zuführen ist, dass dieser bereits 1997 veröffentlicht wurde. Für die geplanten Analysen in dieser Arbeit bietet es sich daher an, Dummynet als Ausgangs- und Bezugspunkt zu nutzen.

Die Trendanalyse in Abb. 3.1 zeigt ebenfalls, dass KauNet inzwischen öfter gesucht wird als Dummynet. Dieser erweitert die Funktionalitäten von Dummynet und eignet sich daher für den direkten Vergleich.

Ein weiterer Vorreiter in der Netzwerkemulation ist NISTNet[13s], welcher jedoch nicht mehr weiterentwickelt wird. Allerdings wurden Teile des Programmkodes von einem anderen Emulator namens NetEm[13q] übernommen. In der Trendanalyse nimmt NetEm die aktuelle Spitzenposition ein, daher wird auch dieser Emulator weiter untersucht. Netem bildet, wie Dummynet, die Basis für andere Emulatoren. Dazu zählt z. B. WANem[13z], welcher ebenfalls in die Auswahl aufgenommen wird. Für diesen Emulator lassen sich genauso wie beim „Linktropy 7500 Pro“ keine direkten Analysen zur Genauigkeit oder Leistung finden, jedoch zeichnet sich WANem durch eine umfangreiche Dokumentation mit Leistungswerten aus.

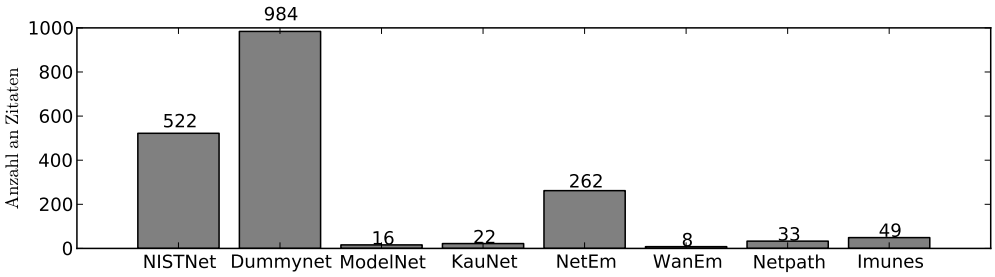


Abbildung 3.2: Anzahl der Zitationen für ausgewählter Emulatoren. (Quelle: [13e])

Abseits der Trendanalyse wird ein weiterer Emulator namens NetPath ausgewählt. Dessen Entwickler betonen, dass NetPath „andere Emulationssysteme deutlich übertrifft“[13r, S. 11].

Die bisher gewählten Emulatoren gehören zur Klasse der NLE (vgl. Abschnitt 2.5). Um ebenfalls Emulatoren aus dem Bereich der VNE analysieren zu können, werden das frei verfügbare ModelNet und Imunes mit in die Auswahl aufgenommen. Letzterer wurde in neueren Arbeiten oft zitiert und verspricht eine sehr einfache Konfiguration von Netzwerktopologien.

Zusammengefasst ergeben sich für die Untersuchungen die folgenden Emulatoren:

- „Linktropy 7500 Pro“ (ohne Analysen)
- DummyNet
- KauNet
- ModelNet
- NetEm
- WANem (ohne Analysen)
- NetPath
- Imunes

3.2 Produktpublikationen und -dokumentationen

Bevor im Einzelnen auf die jeweiligen Analysen bzw. Produktangaben eingegangen wird, werden kurz die Prinzipien und die Umsetzung des Emulators beschrieben. Dadurch ergeben sich die feststehende Merkmale für die Einordnung in die vorgestellte Taxonomie. Soweit Analysen vorhanden sind, werden diese gesondert erläutert und im Fazit zur Betrachtung des jeweiligen Emulators zusammengefasst. Daraus lassen sich u. a. die Vorgehensweise und die verwendeten Werkzeuge für eine Anforderungsanalyse des eigenen Messkonzeptes ablesen. Zusätzlich werden Produktangaben zum jeweiligen Emulator in den Tabellen 3.1 bis 3.9 aufgelistet. Diese Angaben finden sich oft im jeweiligem Handbuch oder werden, soweit vorhanden, aus Messergebnissen der Evaluation abgelesen. Der Umfang dieser Informationen variiert zwischen den Emulatoren und zeigt somit ebenfalls an welchen Stellen noch Analysen benötigt werden.

Beim Hardware-Emulator und WANem sind keine Analysen auffindbar, aus diesem Grund werden lediglich die Produktangaben für spätere Vergleiche in einer Tabelle aufgelistet.

DummyNet ist ebenso wie NetEm Grundlage für andere Emulatoren. Daher werden die Funktionsweisen von Beiden eingehender erläutert. Die Entwickler von KauNet haben diesen ausführlich evaluiert. Weshalb dieser bzgl. der Analysen umfassender betrachtet wird.

3.2.1 Linktropy 7500 pro

Dieser Hardware-Emulator bildet ein WAN zwischen zwei Netzwerkbereichen nach und gehört daher zur Klasse der NLE. Für die Nachbildung stehen dem Versuchsdurchführenden eine Vielzahl von Netzwerkeffekten und Einstellmöglichkeiten zur Verfügung. Dazu gehören beispielsweise C und $\Delta\mathrm{OWD}^j$. Darüber hinaus lassen sich Wahrscheinlichkeiten für die einzelnen Effekte vorgegeben. Ein Software zur Analyse und Aufzeichnung von Verlusten sowie Verzögerungen in Realumgebungen ist ebenfalls beigelegt. Die Aufzeichnungen lassen sich im Anschluss auf dem Emulator abspielen und man erhält so die Netzwerkkonditionen einer bestimmten Realumgebung.

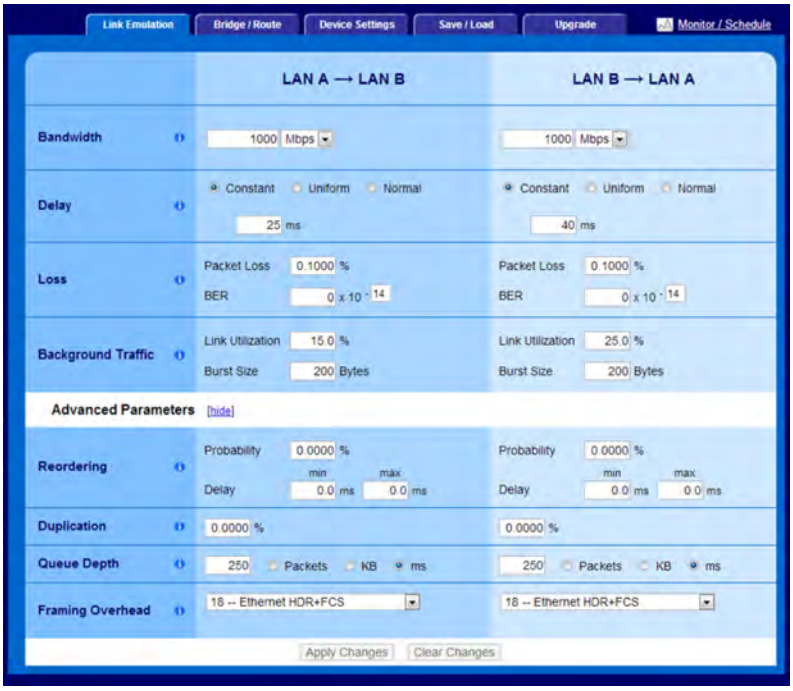


Abbildung 3.3: Weboberfläche des „Linktropy 7500 pro“ Hardware-Emulators.

Gesteuert wird der *Linktropy* über eine Weboberfläche, welche in Abbildung 3.3 zu sehen ist. Aus dieser ist auch eine genaue Auflistung der möglichen Pfadmanipulationen ersichtlich. Für jeden Effekt stehen Informationen bereit, die durch ein Betätigen des Informationssymbols neben dem Effektnamen aufgerufen werden können. Andere Quellen für Genauigkeits- und Leistungsangaben sind der „Hardware Guide“[Tec11] und das Benutzerhandbuch[Tec13].

Produktangaben Der Hardware-Emulator besticht im Vergleich zu allen anderen Software-basierten Emulatoren durch seine Leistung. Dieser kann bis zu drei Millionen Pakete pro Sekunde verarbeiten und arbeitet gleichzeitig mit Paketverzögerungen im Millisekundenbereich. Alle Produktangaben sind in Tabelle 3.1 zusammengefasst.

3.2.2 DummyNet

Die erste wissenschaftliche Publikation zu DummyNet stammt aus dem Jahr 1997 [Riz97]. Rizzo L. beschreibt darin wie mit weniger als 300 Zeilen Quellkode der Kernel von FreeBSD dahingehend angepasst werden kann, dass Pakete zwischen der IP- und der TCP-Verarbeitungsschicht verzögert, verworfen oder neu geordnet werden können. Dazu werden im Protokollstapel zwischen den betreffenden Schichten zusätzliche Warteschlangen mit speziellen Abarbeitungsstrategien (in [Riz97] als *routers* bezeichnet) und Kommunikationsschnittstellen (als *pipes* bezeichnet) mit einer definierten Verzögerung sowie Datenrate (engl. *Bits Per Second* (BPS)) implementiert. In Abbildung 3.4 ist dieser Aufbau dargestellt.

Die Pakete gelangen zunächst in die *routers*, welche eine begrenzte Anzahl an Paketen aufnehmen können und die Abarbeitungsstrategien (standardmäßig *First In First Out* (FIFO)) anwenden.

		Effektqualität	Angaben
Granularität	Datenrate	C CT	$300\,\mathrm{Bit}/s - 1\mathrm{GBit}/s$ je Richtung $0 - 100\%$ in $0,1\%$ Schritten
	Paketeffekte	PLR PDR PRR BER	$0 - 100\%$ in $0,0001\%$ Schritten
	Verzögerungen	OWD $\Delta\mathrm{OWD}^j$	$0\ ms - 10\ s$ in $0,1\ ms$ Schritten
Leistung	Paketrate		$3\,\mathrm{MPkt/s}$

Tabelle 3.1: Angaben zur einstellbaren Granularität und Leistung des „Linktropy 7500 pro“

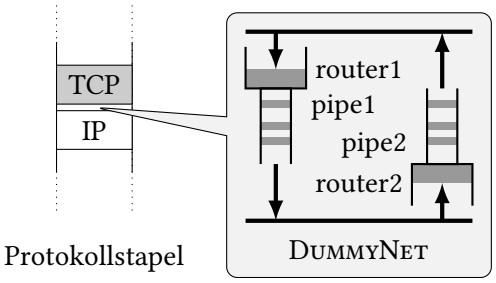


Abbildung 3.4: Funktionsprinzip von DummyNet (nach [Riz97])

Die Abarbeitung kann dabei auch nach anderen Prinzipien erfolgen. An dieser Stelle werden pseudozufällige Paketneuordnung durchgeführt. Dabei wird allerdings kein Zufallsgenerator verwendet, sondern eine feste Zahlenreihe bei der lediglich der Anfangspunkt variiert wird. Anschließend werden die Pakete an die *pipes* übergeben, die jedoch an dieser Stelle festgelegt nach dem FIFO-Prinzip arbeiten. Die Pakete verweilen eine vorgegebene Zeitdauer in diesen *pipes* und werden danach an die nächste Protokollstufe weitergeleitet. In den *pipes* werden die Pakete ebenfalls zufallsähnlich für die benötigte Verlustanzahl verworfen.

Zusätzlich werden in [Riz97] einige Einschränkungen, wie z. B. die Abhängigkeit von der Granularität und Genauigkeit des Systemzeitgeber, verdeutlicht. Diese bestimmt in Interrupt-gesteuerten Systemen die Verarbeitungsrate der *pipes*. Für die Einhaltung von z. B. einer bestimmten Paketverzögerung t_P muss für die Granularität des Zeitgebers T_Z gelten: $T_Z < t_P$

Die erste Version von DummyNet war auf die Verarbeitung von TCP-Paketen unter FreeBSD beschränkt. Die Steuerung von DummyNet geschah über das `sysctl` Kommando. Über die Jahre hinweg fand DummyNet immer mehr Verbreitung und wurde entsprechend weiterentwickelt. In der aktuellen Version 20130607 werden mittlerweile UDP-Pakete und die Betriebssysteme Windows sowie Linux Kernel 3.8 unterstützt. Unter FreeBSD ist DummyNet Bestandteil des Kernels geworden.

Zusätzlich erschien 2010 eine weitere Publikation[CR10] zu DummyNet, welche neben den Angaben zum damals aktuellen Stand des Emulators auch eine Genauigkeits- und Leistungsanalyse enthält.

Aus den Angaben geht hervor, dass die aktuelleren Versionen von DummyNet in der Lage sind mehr als einen Pfadabschnitt nachzubilden. Dafür wurde ein weitere Ebene zur Paketverarbeitung hinzugefügt, welche ankommende und/oder abgehende Pakete klassifiziert, bestimmten *pipes* zuordnet und gegebenenfalls zurück an den Ausgangspunkt der Verarbeitungskette sendet. Diese Schleifenbildung, wie in Abb. 3.5 veranschaulicht, bildet aus Sicht des Paketes eine Verkettung von mehreren *pipes* und damit die verschiedenen Pfadabschnitte mit den jeweiligen Eigenschaften. Diese zusätzliche Ebene wird durch den Einsatz einer angepassten Version von `ipfw`, einem Tool zur Konfiguration der FreeBSD Firewall, ermöglicht. Die Pakete werden daher anhand von Firewall-Regeln klassifiziert und an bestimmte *pipes* weitergeleitet. Dadurch entstehen jedoch zusätzliche Verarbeitungskosten, die im Analyseteil von [CR10] weiter untersucht werden.

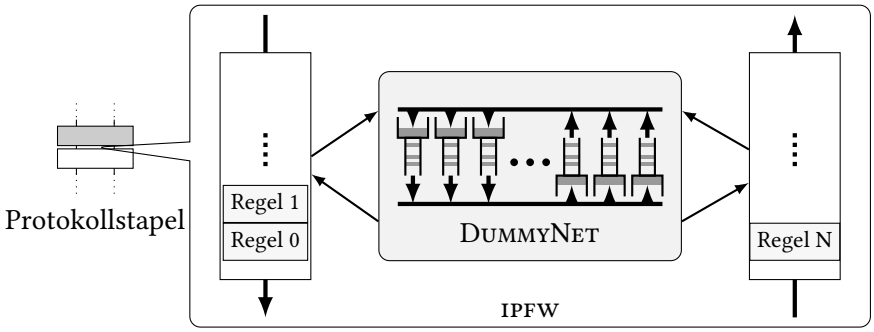


Abbildung 3.5: Funktionsprinzip von DummyNet mit der Erweiterung durch IPFW (nach [CR10])

Der Analyseteil ist zweigeteilt und zeigt neben den Messergebnissen auch die entsprechende Vorgehensweise bei der Genauigkeits- und Leistungsanalyse. Genutzt wird dazu ein einzelner Rechner, dessen technische Merkmale für die entworfene Taxonomie verwendet werden und in Tabelle 3.10 zu sehen sind.

Genauigkeitsanalyse Im ersten Teil der Analyse wird auf die Präzision bzw. auf deren Einflussfaktoren eingegangen. Zum einen bestimmt der Umfang des Netzwerkmodels und zum anderen die umsetzbare zeitliche Koordinierung (engl. *timing*) den Grad der Präzision. Je detaillierter die Emulation stattfinden soll, desto höher ist der Verarbeitungsaufwand. Daher verzichtet DummyNet zum Beispiel auf die Nachbildung von Bitfehlern in Paketen. Aus diesem Grund approximieren auch viele andere Emulatoren diese Effekte.

Laut [CR10] wird der *timing* Aspekt im Wesentlichen durch den konkurrierenden Paketverkehr (engl. *Cross-Traffic* CT) bei mehreren emulierten Knotenpunkten, die Interferenz durch andere Systemprozesse und die Granularität des Zeitgebers beeinflusst. Tabelle 3.2 zeigt den Grad der jeweiligen Beeinflussung.

Die Werte für den CT beziehen sich in der Tabelle auf eine Paketlänge L von 1500 *Byte* mit einer Datenrate A von 1 $Gbit/s$ und 100 $Mbit/s$. Sobald mehrere Knotenpunkte N mit $(N > 1)$

	Beeinträchtigung
Cross-Traffic	$120 \ldots 1200 \mu s$ pro emulierten Knotenpunkt
Interferenz	$> 30 \mu s$, abh. vom Betriebssytem
Granularität	$25 \ldots 1000 \mu s$, abh. vom Betriebssystem

Tabelle 3.2: *timing*: Grad der Beeinträchtigung durch die genannten Faktoren. (Quelle: [CR10])

emuliert werden, entsteht zwischen diesen ein Wettbewerb um Betriebsmittel und es müssen Wartezeiten hinzugefügt werden. Im schwerwiegendsten Fall muss ein Paket dabei (vgl. [CR10]):

$$T = (N - 1) \cdot \frac{L}{A} \tag{3.1}$$

Sekunden warten.

Der Wert für die Interferenz wurde experimentell mit Hilfe von PING-Tests unter verschiedenen Auslastungen der *Arbeitsbereiche* (vgl. Abschnitt 2.3) und für verschieden viele Firewall-Regeln ermittelt. Als Ausgangswert werden die PING-Antworten eines im Leerlauf (engl. idle) befindlichen Rechners für keine, eine und hundert Regeln gemessen. Anschließend wird der Nutzerbereich komplett ausgelastet und die Antwortzeit ebenfalls für die drei Anzahlen an Regeln ermittelt. Es stellte sich heraus, dass die Auslastung des Nutzerbereichs keinen Einfluss auf die Antwortzeiten hatte. Ganz anders verhielt es sich jedoch bei der kompletten Auslastung des Kernelbereichs. Hierbei entstehen zusätzliche Verzögerungen, welche mit zunehmender Regelanzahl ansteigen.

Der dritte Wert ist abhängig von der eingestellten Taktrate des Zeitgebers. Diese wird in Hz angegeben und bestimmt die Anzahl möglicher Systemoperationen pro Sekunde. Paketverzögerungen werden somit intern auf ein Vielfaches dieser Takte ab- bzw. aufgerundet. Im Falle von DUMMYNET werden oft 10000 bis 40000 Hz veranschlagt, was gleichzeitig bedeutet, dass der Fehler 25 bis 100 μs beträgt.

Leistungsanalyse Im zweiten Teil der Analyse wird auf die Messung der Leistungsfähigkeit eingegangen. Als Metrik wird dabei die Verarbeitungszeit je Paket (engl. Per-packet Processing Time (PPT)) benutzt, welche dem Inversen der Paketrate (engl. Packet Per Second (PPS)) entspricht. Die Verarbeitungszeit setzt sich aus verschiedenen Einzelzeiten zusammen, zu denen DUMMYNET jedoch lediglich die Abarbeitungzeit der Firewall-Regeln und die Verzögerung durch die Emulation beiträgt. Die anderen Zeiten sind Systemabhängig.

Um die Verarbeitungsdauer für beide Teilabschnitte zu messen, werden von einem lokalen Paketgenerator kleinstmögliche UDP-Pakete versendet und je nach Experiment an bestimmten Stellen in der Verarbeitung verworfen. Die Pakete werden vor und nach der Firewall sowie nach der Emulation verworfen. Zusätzlich wird die Dauer der Paketverarbeitung für das gesamte System gemessen und die Anzahl der Regeln sowie die Konfiguration der *pipes* als Einflussfaktor untersucht. Das Schema in Abb. 3.6 stammt aus [CR10] und zeigt die Teilabschnitte sowie die Stellen an denen die Pakete verworfen werden.

Für die Berechnung der Verarbeitungsdauer der Firewall werden die gemessenen Zeiten der Durchläufe mit einer einzeln und mit hundert Regeln verglichen. Die Messungen zur Bestimmung der Verarbeitungsgeschwindigkeit der *pipes* werden zunächst ohne jegliche Beschränkung, dann

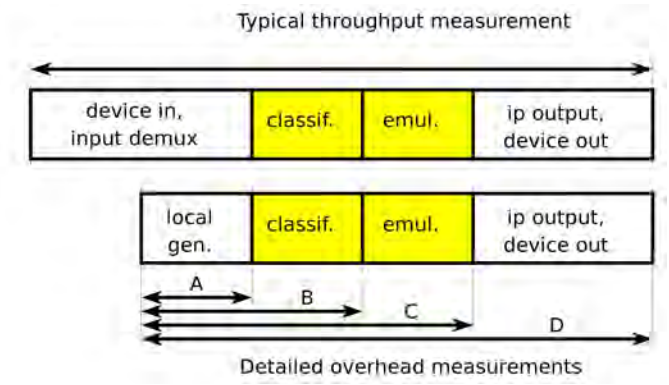


Abbildung 3.6: Oben: die typische Messung der Paketrate durch die gesamte Paketverarbeitung des Betriebssystems. Unten: die verwendete Messmethode, um die Paketklassifikation und Emulation gesondert betrachten zu können. (Quelle: [CR10])

		Effektqualität	Angaben
Granularität	Paketeffekte	PLR PRR	$0 - 100\%$
	Verzögerungen	OWD	Auflösung $0,1$ *ms* (abh. v. Kernel-Taktung)
Leistung	Paketrate		$300\,\mathrm{KPkt/s}$

Tabelle 3.3: Produktangaben zur möglichen Granularität und Leistung von DummyNet (nach [13w])

mit einer bestimmten Verzögerung und abschließend mit einer Beschränkung der Datenrate durchgeführt.

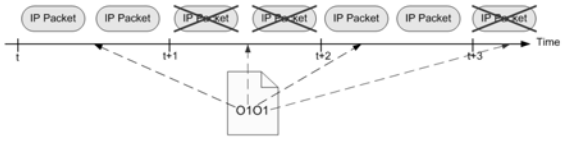
Die Ergebnisse der Messungen liegen für beide Teilabschnitte bei wenigen Mikrosekunden.

Fazit Die vorgestellten Analysen bieten wichtige Anhaltspunkts für ein eigenes Messkonzept. So sollte neben der Wertbestimmung zu einem bestimmten Netzwerkeffekt, auch der *timing* Aspekt untersucht werden. Dadurch lassen sich bereits im Vorfeld der eigentlichen Versuchsdurchführungen Angaben über die Höhe der Abweichungen machen. Des Weiteren spiegelt die PPT im Vergleich zur PPS eher die Leistung des Emulator wieder. Wenngleich die direkte Bestimmung der PPT komplizierter ist und in mehreren Etappen durchlaufen werden muss. Die Angaben zur Genauigkeit und Leistung von DummyNet sind in Tabelle 3.3 zusammengefasst. Zu einigen Effekten, wie z. B. der möglichen Datenrate wurden keine Werte gefunden.

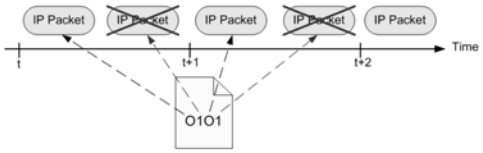
3.2.3 KauNet

KauNet[GHB08] erweitert DummyNet um die Möglichkeiten Bitfehler und Änderungen der Paketverzögerung sowie maximalen Datenrate nachzubilden. Außerdem gestattet KauNet eine präzise Auswahl derjenigen Pakete, welche verworfen bzw. neu geordnet werden sollen. Dazu erfolgt die Selektion entweder zeitgesteuert pro Millisekunde oder datengetrieben je Paket.

Ein besonderes Merkmal von KauNet ist dabei, dass die Selektion als Muster (engl. *pattern*) an den Kernel übertragen wird, um so reproduzierbare Ergebnisse zu erhalten. Die Veranschaulichung in Abb. 3.7 stammt aus einer neueren Dokumentation[Hur+12] zu KauNet und zeigt wie beispielsweise Pakete zeitgesteuert (engl. *time-driven*) oder datengetrieben (engl. *data-driven*) anhand des Musters „0101“ ausgewählt und verworfen werden. Die einstellbare Granularität für Effekte findet, wie in Tabelle 3.4 aufgelistet, daher auch auf Paketebene statt.



(a) Time-driven packet loss



(b) Data-driven packet loss

Abbildung 3.7: (a) Zeit- und (b) datengesteuerte Auswahl bestimmter Pakete durch KauNet anhand des Musters „0101“ (Quelle: [Hur+12])

Anders als bei DummyNet werden Netzwerkeffekte in KauNet deterministisch nachgebildet. Die dafür benötigten Muster werden im Vorfeld einer Emulation durch das beigefügte Tool namens patt_gen generiert und abgespeichert. Anschließend werden diese Muster durch eine angepasste Version des bereits für DummyNet genutzten Kommandos `ipfw` in den Kernelbereich übertragen. Damit diese Vorgaben Wirkung zeigen, erweitert KauNet die *pipes* von DummyNet und fügt beispielsweise innerhalb der *pipes* eine zusätzliche Warteschlange für die Paketneuordnung hinzu. Abbildung 3.8 veranschaulicht Vorgehensweise und die Erweiterung durch KauNet.

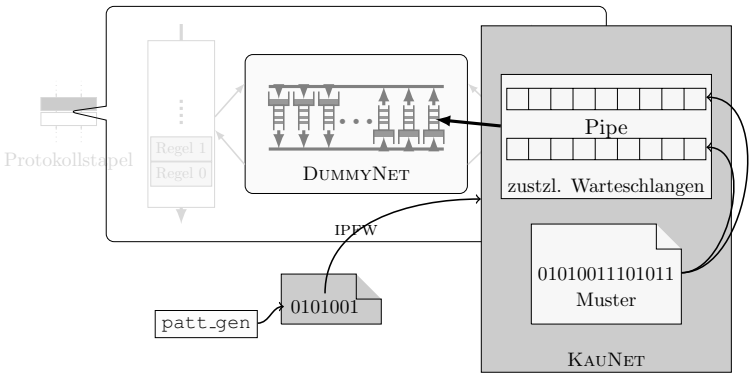


Abbildung 3.8: Funktionsprinzip von KauNet mit dem Mustergenerator `patt_gen`

Um die Dynamik des Versuchsaufbaus weiter zu erhöhen, ist es möglich vor einer Emulation mehrere Muster zu verketten und so eine Wechsel der Vorgaben während der Versuchsdurch-

führung zu ermöglichen. Zusätzlich kann ein Muster auf mehrere *pipes* verteilt werden. Ein Hinzufügen weiterer Muster während der Emulation ist jedoch nicht möglich.

Für KauNet wird in [Hal+12] durch Hall et al. eine umfassende Analyse bzgl. der Leistung und Genauigkeit vorgestellt. Die Untersuchungen finden dafür neben physischen auch in virtuellen und hybriden Versuchsumgebungen statt.

Unter anderem werden in drei Experimenten Analysen zur erreichbaren Durchgangsrate und zur Genauigkeit der Paketverzögerung bei unterschiedlichen Auslastungen durchgeführt. Bevor im Einzelnen auf die Analysen eingegangen wird, soll kurz die Messmethode erläutert werden.

Messmethode Die Werte für die Analysen werden, wie in Abbildung 3.9 dargestellt, an vier verschiedenen Stellen gemessen. Speziell für die Durchgangswerte werden die Datenpakete direkt am Eingang (II) und Ausgang (III) des Emulators erfasst.

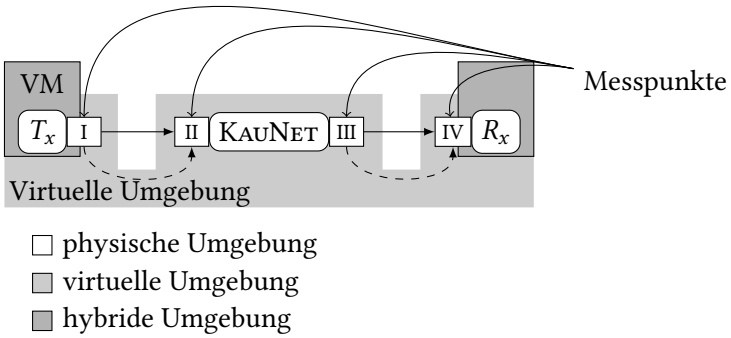


Abbildung 3.9: Die drei Versuchsumgebungen für die Analysen von KauNet.

Für die Erfassung der Werte wird das Werkzeug tcpdump[12] verwendet. Ein anderes Tool namens mgen[13o] dient in den Experimenten als Paketgenerator. Um die Einflüsse durch das Werkzeug zu verringern, verwenden Hall et al. virtuelle Datenträger, sogenannte *Ramdrives*, zur Speicherung der erfassten Paketdaten. Für die virtuelle Umgebung werden zwei verschiedene Testsysteme benutzt. Zum einen wird ein Arbeitsplatzrechner und zum anderen ein Serversystem mit wesentlich mehr Leistung verwendet.

Leistungsanalyse Für den Nachweis der Leistungsfähigkeit von KauNet werden in [Hal+12] zwei Experimente durchgeführt. Beide unterscheiden sich in der Anzahl der verwendeten *pipes*. Im ersten Versuch ist jede der eingesetzten *pipes* für bestimmte Netzwerkeffekte zuständig. Im anderen Experiment dagegen, verfügen mehrere *pipes* über die gleichen Vorgaben. In beiden Fällen wird die mögliche Verarbeitungsgeschwindigkeit des Emulators getestet. Die variablen Faktoren sind dabei die generierte Datenrate, die Paketgröße und speziell im zweiten Versuch die Anzahl der *pipes*.

Bei den Betrachtungen mit mehreren, bzgl. des Effektes, identischen *pipes* bei unterschiedlichen Datenraten und Paketgrößen, wird für jede *pipe* ein Paketstrom erzeugt.

Während der Versuchsdurchführung werden zum einen die Paketverzögerungen am Ein- und Ausgang des Emulators gemessen. Dadurch befinden sich beide Messpunkte auf einer einzelnen

Maschine und eine Uhrensynchronisation ist überflüssig. Zum anderen wird die Verbindung an allen vier Messpunkten (vgl. Abb. 3.9) auf ungewollte Paketverluste geprüft. Wodurch genau die Verluste ermittelt werden können, die tatsächlich durch den Emulator verursacht wurden.

Die physische und hybride Umgebung zeigen die besten Ergebnisse. Die empfangenen Datenraten entsprechen den gesendeten. Einzig beim hybridem Aufbau, bei dem sich lediglich die Endpunkte in einer VM befinden, entstehen wesentlich öfter Paketverluste. Abweichungen zwischen der Empfangs- und Sendedatenrate sind zumeist auf den Paketgenerator MGEN zurückzuführen, da dieser teilweise nicht die geforderte Rate liefert. Daher werden diesbezügliche Auswertungen als Verhältnis zwischen empfangener und der tatsächlich gesendeten Datenrate angegeben.

Für die Betrachtungen mit zunehmender Belastung wird die Verlustrate über der geforderten Datenrate ausgewertet. So zeigt sich die Überlast von KauNet ab einer bestimmten Datenrate und Anzahl an *pipes*.

Genauigkeitsanalyse Die Genauigkeitsbetrachtungen konzentrieren sich auf die Untersuchung der Paketverzögerung bei unterschiedlichen Belastungsstufen. Die *pipes* werden so konfiguriert, dass die Datenrate unbeschränkt bleibt und nur bestimmte Verzögerungen auftreten. Dabei wird die Durchgangsverzögerung in einem ersten Experiment für ansteigende Datenraten mit einer festgelegten Verzögerung gemessen. Das erste Experiment besteht aus mehreren Durchläufen, die auf unterschiedlich festgelegten Verzögerungen basieren.

In einem weiteren Versuch steigt dagegen die Verzögerung und die generierte Datenrate wird mit kleinen Paketen sowie einer geringen Paketrate konstant gehalten. Dadurch sollen alle Einflussfaktoren bis auf die geforderte Verzögerung minimiert werden.

Auch in den Analysen zur Genauigkeit zeigt sich, dass die Messergebnisse innerhalb der virtuelle Umgebungen hinter den erwarteten Werten liegen. So ist die Streuung der unterschiedlichen Verzögerungen deutlich größer als bei den anderen Umgebungen. In beiden Systemen (Arbeitsplatz und Server) zeigen sich ähnliche Werte.

Fazit Die Arbeit von Hall et al. bietet wichtige Anhaltspunkte zum Versuchsaufbau und damit auch zur Anforderungsanalyse in Kapitel 4. So sollte beispielsweise der verwendete Paketgenerator stabil mehrere Paketströme generieren können.

Die Messung der Verzögerungen am Ein- und Ausgang mittels TCPDUMP erhöht zusätzlich die Genauigkeit bei der Wertgewinnung. Zudem sollte die Paketrate bei der Genauigkeitsbetrachtung von Paketverzögerungen gering gehalten werden, um alle systembedingten Verzögerungen zu minimieren.

Die Anzahl von auftretenden Paketverlusten innerhalb des Emulators kann als Maß für dessen Auslastung fungieren und zeigt an, ab welcher Datenrate oder Anzahl an Emulationseinheiten ein Emulator überlastet ist.

3.2.4 ModelNet

Ein anderer Emulator namens ModelNet[Yoc+02] richtet den Fokus auf die Skalierbarkeit und basiert auf einer dezentralen Architektur mit DummyNet als Kernkomponente.

In [Yoc+02] wird gezeigt wie auf Grundlage eines Rechnerverbundes eine hohe Anzahl an Verbindungen mit bestimmen Charakteristiken emuliert werden kann. Dafür wird jedoch auf

		Effektqualität	Angaben
Granularität	Paketeffekte	PLR PRR BER	$0 - 100\%$ für einzelne Pakete
	Verzögerungen	OWD	$0,1$ *ms* (abh. v. Kernel-Taktung)
Leistung	Paketrate		$50\,\mathrm{KPkt/s}$ bei $500\,\mathrm{Byte}$ (abh. v. Paketgröße)

Tabelle 3.4: Angaben zur Granularität und Leistung von KauNet

einen Teil der Netzwerkeffekte verzichtet. Es werden Techniken angewendet, die die vorgegebene Topologie automatisiert vereinfachen, mehrere Endsysteme auf einem einzelnen Rechner virtualisieren und synthetischen Hintergrundverkehr erzeugen.

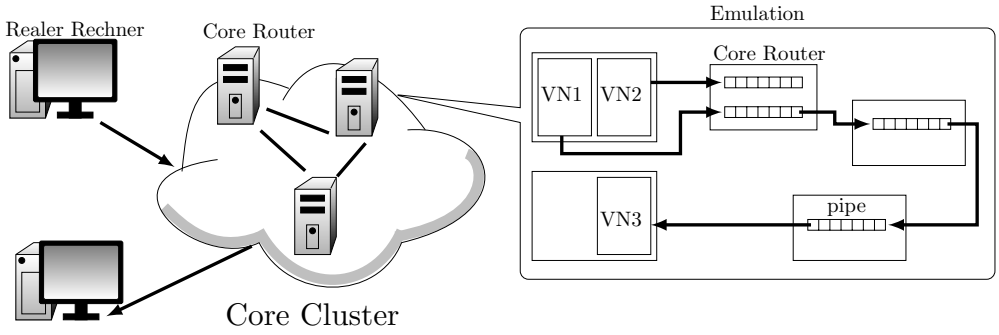


Abbildung 3.10: Schematischer Aufbau von ModelNet am Beispiel mit drei *virtual edge nodes* (VN) auf zwei Realen Rechner und einem *Core Router Cluster* mit drei Rechnern. VN 1 und VN 2 verbinden sich parallel über teilweise unterschiedliche Pipes mit VN 3.

Der Aufbau von ModelNet ist in Abbildung 3.10 schematisch dargestellt. Dieser unterteilt sich in virtuelle Endsysteme (als *virtual edge node* (VN) bez.) und in Server, welche die eigentliche Emulation durchführen und als *Core Routers* bezeichnet werden. Letztere basieren auf DummyNet und arbeiten mit modifizierten FreeBSD-Kerneln. Die Virtualisierung der Endsysteme basiert auf der Zuweisung von mehreren IP-Adressen zu einer einzelnen Netzwerkkarte (IP-Aliasing), Routing-Tabellen und einem Verfahren um Systemkommandos mit eigenen Kommandos zu verknüpfen (Library Interposition). Der Begriff „virtuell“ ist daher nicht ganz passend gewählt.

Die Pfadvorgaben werden mittels Topologie an ModelNet übergeben. Der Emulator unterstützt dafür mehrere Formate, wie z. B. GT-ITM[13m].

Vor einer Emulation wird die vorgegebene Netzwerktopologie in vier Phasen konvertiert, vereinfacht, übersetzt und verteilt. In [Yoc+02] werden diese Phasen als *Create*, *Distill*, *Assign* und *Bind* bezeichnet.

In der *Create*-Phase wird die Topologievorgabe in das GML-Format (graph modeling language) konvertiert und es entsteht ein entsprechender Netzwerkgraph. Anschließend wird in der *Distill*-Phase anhand dieser Graphdefinition eine Menge von *pipe*-Konfigurationen erstellt. Optional

können in dieser Phase Vereinfachungen durchgeführt werden, welche Verarbeitungskosten bzw. Emulationskosten einsparen. In der *Assign*-Phase werden diese Konfigurationen an die *Core Router* verteilt. Als letzten Schritt vor der Emulation, werden in der *Bind*-Phase die VNs erstellt und an die *Core Router* gebunden. Zudem werden an dieser Stelle die zuvor verteilten Konfigurationen der *pipes* umgesetzt.

Die Validation von ModelNet geschieht in zwei Experimenten. Wobei der Versuchsaufbau im ersten Experiment einen einzelnen *Core Router* und im zweiten vier *Core Router* enthält.

Leistungsanalyse Für die Erfassung der Leistungsdaten auf Paketebene entwickelten die Verfasser von [Yoc+02] ein Tool, welches im Kernelbereich arbeitet. Erfasst wird u. a. die CPU-Auslastung und Paketrate (Pakete je Sekunde) bei zunehmender Pfadlänge H und Anzahl gleichzeitiger Verbindungen. Aus Sicht des Emulators wird daher die Länge der *pipe*-Verkettung und die Zahl der parallel betriebenen Ketten erhöht. Es zeigt sich, dass ab einer bestimmten Konstellation die CPU-Auslastung 100% beträgt und ankommende Pakete verworfen werden. In [Yoc+02] wird daher der Begriff ModelNet-*Kapazität* verwendet.

Im zweiten Experiment wurde zudem der zusätzliche Paketaustausch zwischen den *Core Router* untersucht, welcher aufgrund der verteilten *pipes* entsteht. Dieser Austausch wird als *Cross-Core Traffic* bezeichnet. Die VNs kommunizieren über eine Stern-Topologie in einer Weise, dass pro Versuchsdurchgang eine bestimmte Prozentzahl an Verbindungen zwei *Core Router* durchlaufen. Gemessen wird die Paketrate in Bezug auf den prozentualen Anteil an *Cross-Core Traffic*.

Genauigkeitsanalyse Für die Betrachtung der Genauigkeit wird zunächst festgehalten, dass der Zeitgebertakt wesentlich ist. Bei einer resultierenden Granularität von $100\,\mu s$ und einer Verkettung von 10 *pipes* ergibt sich so eine Abweichung von 1 *ms*. Speziell ModelNet versucht diese Abweichung mit in die Konfiguration der *pipes* einzubeziehen.

Anschließend werden Untersuchungen bzgl. der *Distill*-Phase angestellt. ModelNet kann Ende-zu-Ende Verbindungen vereinfachen, indem es pro Verbindung lediglich eine einzelne *pipe* anlegt. Dafür wird die minimale Kanalkapazität der ursprünglichen Pfadvorgabe ermittelt und an diese *pipe* übergeben. Die Verzögerung dagegen ist die Summe aller Teilverzögerungen der ursprünglichen Vorgaben. Für die resultierende Verlustrate wird das Produkt aller Einzelraten verwendet.

Gerade für die Untersuchungen von konkurrierenden Verbindungen ist diese Vereinfachung jedoch nicht erwünscht. Daher bietet ModelNet auch andere Abbildungsmöglichkeiten an. Der Nachweis über die Genauigkeit dieser Möglichkeit wird durch Vergleiche mit dem Simulator ns-2[McC+97] erbracht.

Fazit Für dezentral gestaltete Emulatoren bietet es sich an, zunächst alle Emulationseinheiten auf einem einzelnen Rechner zu installieren. Dieser Rechner wird anschließend einem Stresstest unterzogen, bei dem zum einen die Verkettungslänge und zum anderen die Anzahl paralleler Verkettungen erhöht wird. Das Vorgehen ähnelt dem Vorgehen bei KauNet und es werden dabei ebenfalls die CPU-Last, die Paketrate und der Paketverlust gemessen. Erreicht die CPU-Last $100\,\%$ und die Verlustrate steigt, so ist die Kapazität des Emulators erreicht. Der Begriff Kapazität lässt sich auf andere Emulatoren übertragen und kann als Maß für die *Skalierbarkeit* verwendet werden. In einem weiteren Experiment werden die Emulationseinheiten verteilt und es wird

		Effektqualität	Angaben
Granularität	Paketeffekte	PLR PRR	$0 - 100\%$
	Verzögerungen	OWD	$0,1$ *ms* (abh. v. Kernel-Taktung)
Leistung	Paketrate		120 KPkt/s bei einem *hop* (abh. v. *hops*)

Tabelle 3.5: Produktangaben zur Granularität und Leistung von ModelNet

zusätzlich der *Cross-Core Traffic* und die daraus resultierende Ende-zu-Ende Paketrate ermittelt. Die Werte zur Leistung in Tabelle 3.5 zeigen die Ergebnisse der Messung mit nur einem *Core Router*.

Für Genauigkeitsuntersuchungen bei Ende-zu-Ende Verbindungen können die erwähnten Vereinfachungen als Erwartungswerte benutzt werden.

3.2.5 NetEm

Die bisher vorgestellten Emulatoren basieren auf DummyNet und wurden daher hauptsächlich für den Einsatz unter FreeBSD entwickelt. Abseits von DummyNet wird in [Hem+05] eine eigenständige Lösung namens NetEm für den Einsatz unter Linux ab Kernelversion 2.6 präsentiert. Dieser Emulator basiert ebenfalls auf dem Einsatz von Warteschlangen, welche unter Linux *qdisc*[13y] (*queueing discipline*) genannt werden. In der Paketverarbeitung von Linux steht jedem Ein- und Ausgang einer Netzwerkschnittstelle eine Haupt-*qdisc* vor. Mit Hilfe des *Linux traffic control tools* (kurz `tc`) können nach einer Haupt-*qdisc* weitere *qdisc*-Warteschlangen baumartig hinzugefügt werden. Ein Schema dieser Struktur ist in Abbildung 3.11 zu sehen. Die Baumknoten werden *classful qdisc* genannt und unterstützen Paketfilter. Die Blätter dagegen heißen *classless qdisc* und versenden lediglich eingehende Pakete weiter an das zugeordnete Netzwerkinterface.

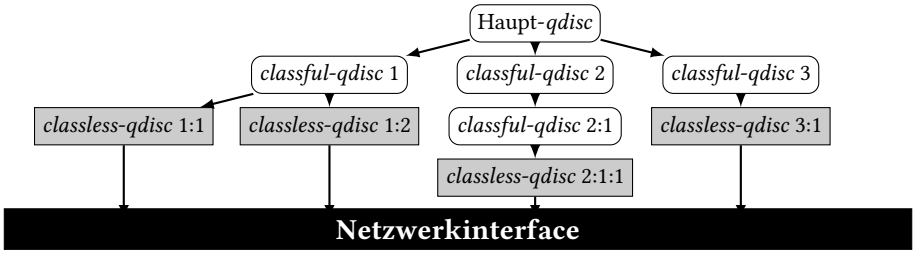


Abbildung 3.11: Baumartige Struktur der *qdisc*-Warteschlangen unter Linux.

NetEm ist bis zur Kernelversion 2.6.29 als *classful qdisc* implementiert. In den späteren Version jedoch, gehört NetEm zu den *classless qdisc*. Daher sind keine Pfadverkettungen wie bei DummyNet mehr möglich. Konfiguriert wird NetEm mittels dem Tool `tc`. Außerdem existiert eine NetEm-GUI [13p], welche die Eingabe vereinfacht. Der Emulator unterstützt Paketverluste, -dopplungen, -neuordnungen sowie -verzögerungen per Häufigkeitsverteilung und kann die maximale Datenrate beschränken. Dafür werden am Eingang der NetEm-*qdisc* die betreffenden

Pakete durch einen Pseudozufallsgenerator ausgewählt und verworfen bzw. verdoppelt. Anschließend erhalten die verbleibenden Pakete einen Zeitstempel mit der Summe aus der aktuellen Zeit und der geforderten Verzögerung. Mit diesem Sendezeitpunkt werden die Pakete in die eigentliche Warteschlange einsortiert und verweilen dort bis die jeweilige Verzögerung abgelaufen ist. Eine Veranschaulichung dieses Arbeitsablaufes ist in Abbildung 3.12 zu finden.

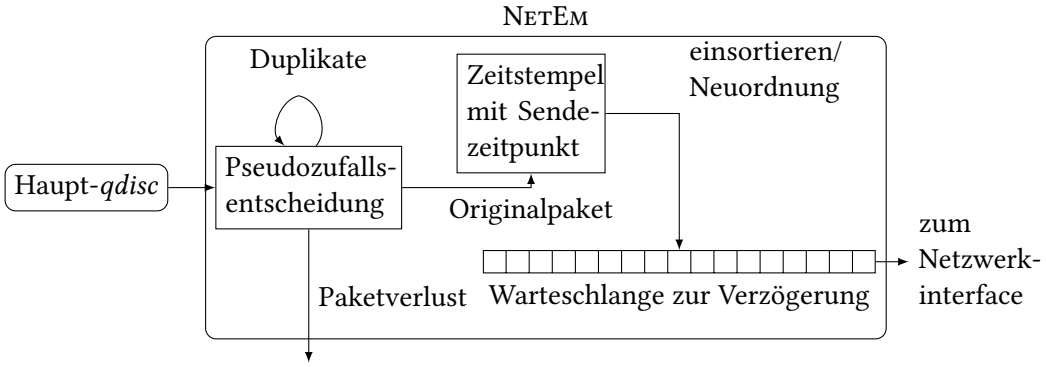


Abbildung 3.12: Schema der Paketverarbeitung durch NetEm.

Der Autor von [Hem+05] beschreibt mit Hinblick auf die Genauigkeit von NetEM, dass dieser durch die Granularität des Zeitgebers, durch den Pseudozufallsgenerator und die Puffergröße der Netzwerkkarte beeinflusst wird. Daher ist, wie bei DummyNet, der Zeitgeber entscheidend für den Grad der Genauigkeit. In vorausgehenden Untersuchungen zur Genauigkeit von DummyNet zeigt sich, dass dessen Zufallsgenerierung ebenfalls problematisch für kurzzeitig Verbindungen ist und die Messwerte weit unter den geforderten Werten liegen. Die Puffergröße muss für die Analyse der Genauigkeit in Betracht gezogen werden, wenn ein hohes kurzzeitiges Paketaufkommen (engl. *Packet-Burst*) erforderlich ist.

Genauigkeitsanalyse Die Evaluation von NetEm fällt in [Hem+05] sehr kurz aus und beruht auf zwei 90-sekündigen TCP-Ströme. Auf eine Leistungsanalyse wurde komplett verzichtet. Der Autor weißt darauf hin, dass NetEm nicht das reale Internet nachbilden kann und es eher wichtig ist einzelne Verbindungen korrekt darzustellen. Daher wurde zunächst eine reale TCP-Verbindung über einen 1 Mbps DSL-Anschluss gemessen und anschließend versucht diese mittels NetEm nachzubilden. Genutzt wird in beiden Fällen ein Verfahren namens *Kprobes*[13k], welches auf Kernelebene bestimmte Instruktionen erfasst und auswerten kann. Die Erfassung verläuft ohne Unterbrechung der betrachteten Instruktionen.

Eine umfangreichere Analyse von NetEm bietet die Arbeit [Jur+11] aus dem Jahr 2011. Zu diesem Zeitpunkt ist NetEm bereits Teil des Kernels von Linux geworden.

Der Versuchsaufbau besteht in [Jur+11] aus drei in Reihe verbundenen Laptops. Als Paketgenerator wird das Tool D-ITG [AP12] genutzt. Untersucht wird die Genauigkeit von NetEm bzgl. der OWD, ΔOWD^j und der Verlustrate. Die benötigten Werte werden mit dem Messwerkzeug QosMet[Mic04] erfasst. Die für die Ermittlung der OWD notwendige Uhrensynchronisation wird durch den Einsatz von GPS umgesetzt.

Vor einer Messung wird sichergestellt, dass alle nicht benötigten Prozesse beendet wurden. Außerdem wird zunächst die Grundverzögerung gemessen, welche ohne den Einfluss von NetEm

		Effektqualität	Angaben
Granularität	Paketeffekte	PLR PDR PRR BER	$0 - 100\%$ $\frac{1}{2^{32}} = 0,0000000232\%$ Schrittweite
	Verzögerungen	OWD ΔOWD^j	$0,1$ *ms* (abh. v. Kernel-Taktung)

Tabelle 3.6: Angaben zur Granularität von NetEm (nach [13q])

existiert. Der Mittelwert der Grundverzögerung wird in den folgenden Messungen herausgerechnet. Für jede der drei genannten Metriken werden in mehreren Versuchsdurchgängen nacheinander aufsteigende Werte vorgegeben. Jeder Durchgang umfasst eine bestimmte Anzahl an Probepaketen. Die Genauigkeit ergibt sich aus den Abweichungen der gemessenen von den vorgegebenen Werten. Die Ergebnisse zur Paketverzögerung zeigen, dass NetEm erst ab einer Verzögerung von 50 *ms* mit einer Genauigkeit von $99\,\%$ arbeitet.

Eine Leistungsanalyse entfällt auch in dieser Arbeit. Entsprechend werden in Tabelle 3.6 lediglich die Angaben zur möglichen Granularität gelistet.

Fazit Die Nachbildung real existierender Verbindungen ist eine Möglichkeit die Genauigkeit zu messen. Dabei steht nicht nur eine einzelne Effektqualität im Fokus der Untersuchung, sondern es wird eine Kombination aus mehreren betrachtet. Als Messwerkzeug kann Kprobes verwendet werden, welches transparent gegenüber der Emulation arbeitet. Eine andere Möglichkeit ist, ein Ende-zu-Ende Messtool wie QosMet zu verwenden. Auch in diesem Fall wird der Emulationsprozess nicht beeinflusst. Ein Paketgenerator wie D-ITG bietet viele variierende Paketströme mit unterschiedlichen Paketabständen und -größen. Mit GPS ist man in der Lage eine Uhrensynchronisation durchzuführen und so präzisere Messwerte für die OWD zu erhalten.

Um den Einfluss des Versuchsaufbaus auf die Emulation abzuschätzen, sollte im Vorfeld eines Versuchsdurchlaufs eine Messung der geforderten Werte ohne Emulator durchgeführt werden. Der Messbereich der anschließenden Untersuchungen sollte möglichst groß gewählt werden, um so den Arbeitsbereich des Emulators bestimmen zu können.

3.2.6 WANem

Ein Emulator, welcher eine besonders einfache Handhabung verspricht, ist WANem[KN11]. Dieser ist als NLE konzipiert und stellt sich die Aufgabe ein WAN zwischen zwei Testsystemen nachzubilden.

Der Emulator wird als „Live-CD“ sowie als „VMWare-Image“ zum Download angeboten und bedarf daher keiner Installation bzw. Anpassung eines existierenden Betriebssystems. WANem kann, da dieser auf NetEm aufbaut, über `tc` und/oder einer angepassten Version der NetEm-GUI konfiguriert werden. Allerdings gestattet WANem mehr Konfigurationsmöglichkeiten und kann zusätzlich zu den *Effektqualitäten* von NetEm auch *Unterbrechungen* (vgl. Abschnitt 2.4.4) nachbilden.

		Effektqualität	Angaben
Granularität	Datenrate	C	$120\,\mathrm{KBit}/s - 50\mathrm{MBit}/s$
	Paketeffekte	PLR PDR PRR BER	$0 - 100\%$ $\frac{1}{2^{32}} = 0,0000000232\%$ Schrittweite
	Verzögerungen	OWD $\Delta\mathrm{OWD}^j$	Die minimale Auflösung beträgt 10 *ms*
	Unterbrechungen	Idle, Zufällig	Die minimale Auflösung beträgt 1 *s*
Leistung	Paketrate		10000 Pkt/s (100 MBit-Netzwerkkarte)

Tabelle 3.7: Produktangaben zur Granularität und Leistung von WANem (nach [KN08])

Die Implementation der Unterbrechungen wird in einer zusätzlichen Arbeit[NK11] beschrieben und beruht im Wesentlichen auf dem Einsatz von `iptables` und einem Kernel-Modul zur Überwachung von TCP-Verbindungen namens `conntrack`.

Produktangabe Für den Vergleich mit eigenen Messergebnissen werden an dieser Stelle die Angaben zur Genauigkeit und Leistungsfähigkeit aus der Dokumentation[KN08] des Emulators aufgezeigt. Die Werte beruhen zum Teil auf theoretischen Betrachtungen oder werden direkt vorgegeben.

WANem unterstützt die Nachbildung von Paketverlusten, -dopplungen, -neuordnungen und -verzögerungen. Letztere können u. a. Pareto-verteilt variieren und korrelieren. Zudem können Bitfehler und, wie bereits erwähnt, Unterbrechungen emuliert werden. Zur besseren Übersicht werden diese erreichbaren Werte als Tabelle in 3.7 dargestellt.

3.2.7 Imunes

Ein ganz eigenes Konzept setzt der Emulator Imunes[ZM04] um. Der Name steht für *Integrated Multiprotocol Network Emulator/Simulator* und soll andeuten, dass Imunes sowohl IPv4- als auch IPv6-Verkehr emulieren kann und diverse Routing-Protokolle unterstützt. Die Grundlage dieses Emulators bilden Netzwerktopologien, welche mit Hilfe „leichtgewichtiger“ Virtualisierungen unter FreeBSD umgesetzt werden. „Leichtgewichtig“ bedeutet dabei, dass die Bausteine einer Topologie mit Hilfe von klonbaren Netzwerkstapeln und privaten Prozessen virtualisiert werden. Jeder dieser Stapel besitzt eigene Ressourcen wie z. B. Routentabellen oder Netzwerkschnittstellen. Die privaten Prozesse wiederum umfassen beispielsweise exklusive Zugriffe auf die CPU, welche mittels *BSD jail*[13d] ermöglicht werden. Pakete innerhalb der Topologie werden durch Zeiger-Referenzen weitergeleitet und nicht wie üblich durch Kopien. Dadurch wird die Leistungsfähigkeit von Imunes deutlich erhöht. Abbildung 3.13 stammt aus einer neueren Veröffentlichung[PM06] zu Imunes und veranschaulicht diese Art der Virtualisierung mit Hilfe eines Vergleichs zur traditionellen Art. In dieser Publikation wird außerdem eine Erweiterung vorgestellt, welche es

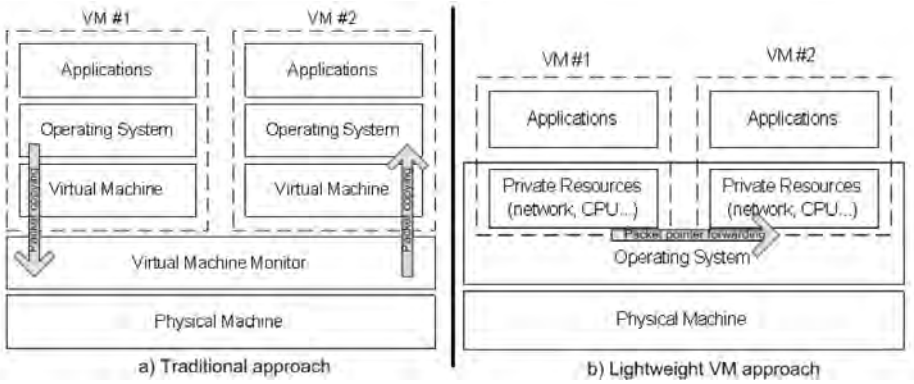


Abbildung 3.13: Leichtgewichtige Virtualisierung unter IMUNES (Quelle: [PM06])

ermöglicht, IMUNES *dezentral* zu betreiben. Dazu werden mehrere Rechner über ein peer-to-peer Netzwerk zusammengeschlossen.

Die Topologien werden mit Hilfe einer TCL/TK basierten GUI erstellt. Wobei zwischen Knoten (engl. *nodes*) und Verknüpfungen (engl. *links*) unterteilt wird. Die Knoten werden weiter unterschieden in *link layer nodes* und in *network layer nodes*. Erstere umfassen alle diejenigen Netzwerkbausteine, die Pakete ohne bestimmte Kontrollen weiterleiten (Switch, Hub). Letztere bilden die Endsysteme sowie Router nach und verfügen über die komplette Funktionalität eines FREEBSD-Systems. D. h., auf diese *network layer nodes* können eigenständige UNIX-Konsolen geöffnet und Programme ausgeführt werden. Dadurch sind u. a. Routenänderungen während der Emulation möglich.

Die Verknüpfungen verbinden zwei Knoten und können bestimmte Effektqualitäten, wie z. B. die maximale Datenrate oder BER, emulieren. Eine konkrete Auflistung der Effekte mit ihren Bereichsgrenzen ist in Tabelle 3.8 zu finden. Die Werte stammen aus dem Benutzerhandbuch[Uni12] zu IMUNES.

Leistungsanalyse In der ursprünglichen Arbeit[ZM04] zu IMUNES stehen zum einen die Auswirkung der Virtualisierung und zum anderen die Anzahl der möglichen Pfadabschnitte (engl. *hops*) im Mittelpunkt der Betrachtungen.

Die Auswirkung wird durch einen Vergleich mit einem unmodifizierten System überprüft. Gemessen wird dazu die erreichbare BTT für verschiedenen Paketgrößen.

Die mögliche Pfadlänge wird ebenfalls durch die BTT, jedoch in diesem Fall für aufsteigende Anzahlen von *hops*, gemessen. Gleichzeitig wird die Auslastung der CPU dokumentiert und mit der jeweiligen Anzahl an *hops* verglichen. Eine Genauigkeitsanalyse wurde in [ZM04] nicht durchgeführt.

Für die *dezentrale* Erweiterung von IMUNES haben dessen Entwickler eine eigenständige Untersuchung[PPM08] veröffentlicht. In dieser wird ebenfalls auf eine Analyse der Genauigkeit verzichtet und es wird vielmehr auf die Skalierbarkeit, Antwortzeit, Verarbeitungszeit und den Durchsatz (TCP) eingegangen.

Die Messungen zur Skalierbarkeit und Antwortzeit werden dazu auf einer Maschine innerhalb des Rechnerverbundes durchgeführt, welche neben den eigentlichen Emulationsaufgaben noch

zusätzliche Aufgaben erfüllt. Diese beinhalten die Verteilung der Nutzervorgaben an die anderen Verbundteilnehmer sowie den Start der Emulation.

Für die Skalierbarkeit werden die Auslastungen der CPU und des Arbeitsspeichers bei einer aufsteigenden Anzahl an Netzwerkelementen und Verbundrechnern gemessen. Dabei zeigen sich deutliche Unterschiede zwischen der CPU- und der Arbeitsspeicherauslastung. Während die CPU kaum Reaktionen zeigt, sind die Werte des Arbeitsspeichers für steigende Verbundteilnehmer und Netzwerkelementen stark verschieden.

Die Antwortzeit bezieht sich auf die Wartezeit zwischen dem Signal zum Start und dem tatsächlichen Beginn einer Emulation. Dazu werden auch hier die Anzahl der Verbundteilnehmer erhöht und für unterschiedlich viele Netzwerkknoten gemessen. Es zeigt sich, dass sich ab einer bestimmten Anzahl an Verbundrechnern kein Zeitgewinn mehr erreichbar ist.

Für die Verarbeitungszeit wird zunächst die RTD für eine reale Verbindung mittels `ping` gemessen. Diese Umlaufverzögerung wird als t_{real} bezeichnet. Der Versuchsaufbau besteht dazu aus zwei Rechnern, welche direkt miteinander verbunden sind. Anschließend wird dieser Aufbau mit IMUNES auf einem der beiden Rechner nachgebildet. Es wird ebenfalls die RTD gemessen und als t_{imun} festgehalten. Danach werden beide Endsysteme auf unterschiedlichen Rechnern virtualisiert, die Umlaufverzögerung gemessen und als t_{tot}. Nach [PPM08] ergibt sich die Verarbeitungszeit t_{proc} mit Hilfe der folgenden Gleichung:

$$t_{proc} = t_{tot} - (t_{imun} + t_{real})$$

Der Durchsatz von TCP-Paketen wird ebenfalls erst in einer realen Umgebung, dann in einer zentralen und abschließend in einer dezentralen Emulation mittels NETPERF gemessen. Den höchsten Durchsatz bietet die zentrale Emulation, gefolgt vom realen Aufbau und der dezentralen Lösung.

Fazit Der Vergleich zwischen einem unmodifizierten System bzw. einer realen Versuchsumgebung und einer für die Emulation angepassten Umgebung gibt Aufschluss über den Einfluss des Emulators. Die gemessene Einflussnahme bezieht sich dabei auf den eingesetzten Aufbau und kann Tendenzen für andere Versuchsumgebungen liefern. Es zeigt sich für die Untersuchungen zur Systemauslastung durch IMUNES, dass vor allem der Arbeitsspeicherbedarf Hinweise zur Leistungsfähigkeit liefert.

Die Antwortzeit des Emulators ist besonders bei dezentralen Lösungen eine mögliche Größe zur Beurteilung.

Die vorgestellte Messung der Verarbeitungszeit und des Durchsatzes kann abgewandelt auch für andere Emulatoren verwendet werden. Eine direkte Verwendung dieser Messweise würde voraussetzen, dass ein Emulator sowohl zentral als auch dezentral verwendet werden kann.

3.2.8 NetPath

Die Entwickler des Emulators NETPATH beschreiben diesen in [IHL07] als extrem leistungsfähig und präzise. In einer umfassenden zweiteiligen Evaluation zeigen sie, dass die erreichbaren Werte weit über denen anderer Software-basierter Emulatoren liegen und nahe an die eines Hardware-basierten Produktes heranreichen. Der Umfang der emulierbaren Effekte hebt NETPATH

		Effektqualität	Angaben
Granularität	Datenrate	*C*	$0\,\text{Bit}/s - 1\text{GBit}/s$
	Paketeffekte	BER	$0 - 10^{-12}$ Verhältnis
		PDR	$0 - 50\%$ in 1% Schritten
	Verzögerungen	OWD	$0 - 10^{7}\ \mu s$

Tabelle 3.8: Produktangaben zur Granularität von IMUNES (nach [Uni12])

ebenfalls deutlich von anderen Emulatoren ab. Bis auf die Erzeugung von CT deckt sich dessen Effekteumfang mit dem des LINKTROPY.

Des Weiteren werden in dieser Arbeit Begriffe wie *Raw Performance* (dt. Rohleistung) und maximale verlustfreie Paketrate (original engl. Bez. *maximum loss-free packet forwarding rate* (MLFFR)) geprägt. Letzterer ist für das eigene Messkonzept interessant und wird im folgenden Abschnitt zur Leistungsanalyse näher beschrieben. Zunächst sollen jedoch die Funktionsprinzipien von NETPATH umrissen werden.

Um diese, laut [IHL07], hohe Präzision zusammen mit einer extremen Leistungsfähigkeit zu erreichen, wählen die Entwickler von NETPATH eine Plattform namens CLICK MODULAR ROUTER[Koh+00]. Mit dieser ist es möglich eine auf *Polling*-basierte, statt eine auf *Interrupt*-basierte, Paketabfrage an der Netzwerkkarte durchzuführen. Dadurch wird schneller auf ein ankommendes Paket reagiert. Außerdem ist CLICK modular aufgebaut und ermöglicht es, durch das Zusammensetzen verschiedener Basismodule, ein Element mit den geforderten Eigenschaften zu kreieren. Die Konfiguration der Module bzw. Elemente wird mit einer deklarativen Sprache durchgeführt, die zudem Abstraktionsmöglichkeiten für zusammengesetzte Elemente bietet. Solche Elemente bilden beispielsweise sogenannte *PollDevices*, welche die Netzwerkkarte immerwährend auf eingehende Pakete abfragt und bei positiver Antwort diese Pakete sofort mit Zeitstempeln versieht.

Andere Elemente ermöglichen NETPATH als Multi-Link Emulator zu fungieren. Eingehende Pakete werden daher nach bestimmten Merkmalen klassifiziert und so an verschiedene nachfolgende Elemente verteilt.

Die Entwickler sehen jedoch bei manchen Elementen weitere Verbesserungsmöglichkeiten und passen u. a. Warteschlangenelemente an. So entstehen neue Elemente mit verbesserten Leistungswerten, welche im ersten Teil der Evaluierung aufgezeigt werden.

Leistungsanalyse Die Evaluation geschieht in zwei Teilen. Im ersten Teil werden die *Rohleistungs*charakteristiken von NETPATH gemessen. Diese umfassen den Durchsatz, die Skalierbarkeit und Präzision. Da Emulatoren ab einer bestimmten Datenrate anfangen Pakete zu verwerfen und so Leistungsdaten verfälscht werden, stehen in [IHL07] vor allem verlustfreie Messungen der *Rohleistungen* im Mittelpunkt. Beispielsweise wird die Verarbeitungsleistung von NETPATH für verschiedene Paketgrößen und -raten immer im Zusammenhang mit einer Verlustfreien Übertragung betrachtet.

Die Rate, bis zu der eine solche Übertragung möglich ist, wird in [IHL07] als die maximal erreichbare verlustfreie Paketverarbeitungsrate (kurz MLFFR) bezeichnet.

		Effektqualität	Angaben
Granularität	Verzögerungen	OWD ΔOWD^j	Die minimale Auflösung beträgt 1 *ms*
Leistung	Paketrate		300 KPkt/s (MLFFR)

Tabelle 3.9: Angaben zur Granularität und Leistung von NetPath (nach [Aga05])

Des Weiteren werden im ersten Teil der Evaluation hauptsächlich die Verbesserung der angepassten Elemente gezeigt. Die Präzision der modifizierten Warteschlangen wird z. B. anhand einer eingestellten Paketverzögerung sowie mit Hilfe aufsteigender Paketraten und unter Beachtung der MLFFR nachgewiesen. Dabei wird die Abweichung der gemessenen zur eingestellten Verzögerung je Paketrate ausgewertet. Für den Nachweis der möglichen Skalierbarkeit wird die MLFFR für eine aufsteigende Anzahl an emulierten Pfadabschnitten gemessen. Die erreichte MLFFR ist in Tabelle 3.9 zu finden, welche zusätzlich auch die mögliche Granularität bei der Vorgabe der Paketverzögerung enthält.

Der zweite Teil beinhaltet Vergleiche mit anderen Emulatoren. Ausgewählt werden dafür ein Hardware-Emulator namens „Adtech SX-14“, NISTNet, DummyNet und ModelNet. Insgesamt werden drei Experimente durchgeführt. Die Paketgröße wird für alle Durchgänge auf 66 Byte festgelegt.

Das erste Experiment testet die Emulatoren auf die Einhaltung einer vorgegebenen Paketverzögerung bei einer ansteigenden Paketrate. Im zweiten Experiment werden die auftretenden Paketverluste als Funktion der Paketrate betrachtet. Zu Beachten ist, dass die MLFFR in beiden Experimenten außen vor gelassen wird.

Im letzten Experiment werden die Antwortzeiten von mehreren TCP-Verbindungsanfragen bei einer eingestellten Umlaufverzögerung und mit steigender Paketrate gemessen. Gleichzeitig werden auftretende Paketverluste registriert und mit in die anschließende Betrachtung einbezogen. Der Hardware-Emulator dient in der Auswertung als Referenz. Die Werte werden als Häufigkeitsverteilung relativ zur Referenz aufgetragen und zeigen z. B., dass NetPath als einziges dicht an den Werten des Hardware-Emulators liegen.

Fazit Die Evaluation in dieser Publikation enthält den wichtigen Hinweis, dass bei der Messung von Leistungsgrenzen immer der Paketverlust betrachtet werden sollte. Eine Metrik wie die vorgestellte MLFFR sollte für ein eigenes Konzept mit in Betracht gezogen werden. Durch diese lassen sich beispielsweise Aussagen zur Skalierbarkeit eines Emulators erzeugen.

Die Untersuchung der Einhaltung einer festgelegten Vorgabe für eine variierende Belastung ermöglicht es, eine Skala für die Genauigkeit in Abhängigkeit zu dieser Last zu erstellen.

Einen direkten Vergleich zwischen verschiedenen Emulatoren gestattet die Messung ungewollter Paketverluste bei steigender Datenrate. Einen Hardware-Emulator als Referenzpunkt zu nutzen, kann für eigene Untersuchungen nicht in Betracht gezogen werden, da sowohl Software als auch Hardware-Emulatoren geprüft werden sollen. Die Darstellung der Ergebnisse als Häufigkeitsverteilung mit Referenzpunkten gestatten einen schnellen Blick auf die Präzision der Emulation.

3.3 Vergleichsstudien

In diesem Abschnitt werden die Vergleichsstudien [NR09], [Sha+10] und [Min+11] bezüglich ihrer Messmethodik und Ergebnisse analysiert. Die beiden letztgenannten Studien stammen von den gleichen Autoren (Minhas et al.) und wurden in den aufeinanderfolgenden Jahren 2010/2011 mit unterschiedlichen Schwerpunkten veröffentlicht. Während in [Sha+10] die Stabilität der Paketverzögerung untersucht wird, steht in [Min+11] die Präzision bei der Begrenzung der Datenrate im Vordergrund.

In der ersten Studie [NR09] von Nussbaum et al. werden beide Größen im Zusammenhang mit dem vom Emulator verwendeten Zeitgeber betrachtet.

3.3.1 Nussbaum, L. und Richard, O. (2009)

Nussbaum et al. untersuchen in [NR09] die drei Emulatoren NISTNet, NetEm sowie DummyNet bzgl. ihrer *Funktionsweise* und Genauigkeit bei der Emulation von Verzögerungen sowie Datenraten. Für das eigene Messkonzept sind vor allem die Vorgehensweisen und Vorbetrachtungen interessant. Die Funktionsweisen von DummyNet und NetEm wurden bereits in den vorhergehenden Abschnitten erläutert. Einzig für NISTNet sei zu erwähnen, dass dieser als Zeitgeber die physikalische Uhr (engl. *real-time clock*) mit einer Taktung von 8192 Hz verwendet. Diese Erwähnung ist wichtig, da auch in [NR09] auf die hohe Bedeutung des Zeitgebertaktes für die Genauigkeit des Emulators hingewiesen wird. Die erste Betrachtung bezieht sich daher zunächst auf unterschiedliche Taktungen bei einer fest vorgegebenen Paketverzögerung. Anschließend werden Messung zur emulierbaren Datenrate, welche ebenfalls auf Verzögerungen beruht, durchgeführt. Die genaue Vorgehensweise dieser Genauigkeits- und Leistungsanalyse sowie der Versuchsaufbau werden im Folgenden beschrieben.

Versuchsaufbau Die Versuchsumgebung besteht, wie in Abb. 3.14 veranschaulicht, aus einem Switch mit zwei Endpunkten und einem Rechner, der als Router für die beiden Endpunkten fungiert. Auf diesem Router werden die drei Emulatoren installiert und nach der Feststellung der Grundverzögerung sowie maximal möglichen Datenrate für den jeweiligen Versuchsdurchgang ausgeführt.

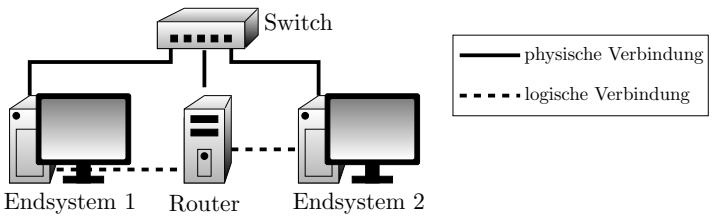


Abbildung 3.14: Versuchsaufbau in [NR09].

Für die Betrachtungen der unterschiedlichen Taktungen bei einer festgelegten Verzögerung wird die RTD als Metrik und eine angepasste Version von ping als Messwerkzeug gewählt. Die Datenrate dagegen wird mittels BTT und dem Werkzeug iperf untersucht.

Genauigkeitsanalyse Der dominierende Faktor für eine präzise Emulation von z. B. Verzögerungen ist der zugrundeliegende Zeitgeber bzw. dessen Taktung. In [NR09] wird dieser Einfluss pro Betriebssystem und Kernelversion für drei verschiedene Taktungen gemessen. Für DummyNet wird beispielsweise die FreeBSD Version 6.1 und 7.0 mit den Taktungen 100 Hz, 1 KHz sowie 10 KHz verwendet.

Gemessen werden die RTD von ICMP-Paketen für alle drei Taktungen bei einer eingestellten Paketverzögerung von 10 *ms*. Die Idee dazu beruht auf der Annahme, dass bei einer hohen Anzahl von Paketen, welche in kurzen Abständen den Emulator passieren, Variationen zwischen den einzelnen RTD zu messen sind, sobald die Sendefrequenz höher als die Taktung des Zeitgebers ist. Der Grund dafür ist, dass der Emulator Pakete nur im Takt des Zeitgebers verarbeiten kann. Abhängig von der Implementierung werden die Pakete daher leicht verzögert oder verfrüht weitergeleitet.

Standardmäßig verwendet ping einen Zeitgeber mit einer für die Untersuchungen zu geringen Taktung. Nussbaum et al. wechselten daher dessen Zeitgeber, sodass auch Pakete mit einer Frequenz von bis zu 20 KHz versendet werden können.

Die Ergebnisse werden in Form von Diagrammen zum einen als gemessene RTD über der betrachteten Zeit und zum anderen als Häufigkeitsverteilung ebenfalls über der Messdauer präsentiert. Ein Beispiel dieser Diagramme ist in Abbildung 3.15 zu sehen.

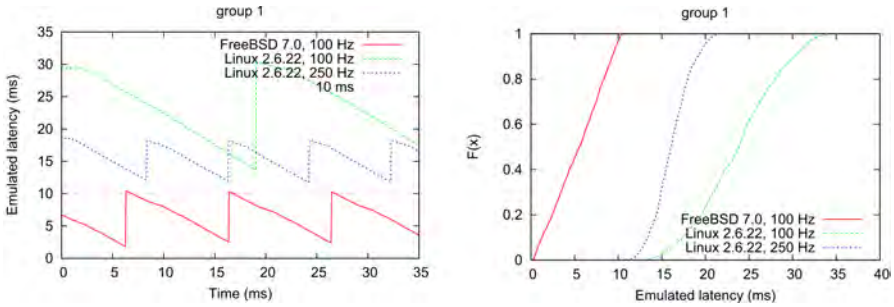


Abbildung 3.15: Messergebnisse: Einfluss der Taktrate auf die Genauigkeit bei der Emulation einer Paketverzögerung von 10 *ms*. (rechts: Häufigkeitsverteilung, Quelle: [NR09])

Erstere zeigen für eine geringe Taktung ein Sägezahnmuster, was darauf zurückzuführen ist, dass eingehende Pakete beim Emulator zunächst zwischengespeichert und erst mit dem nächsten Takt weiterverarbeitet werden. Die Darstellung der Häufigkeitsverteilung gestatten einen Überblick auf die Streuung und die Genauigkeit. Je dichter die Werte einer Kurve an den vorgegebenen Wert von 10 *ms* heranreichen, desto präziser sind die emulierten Paketverzögerungen.

Um die Präzision zu verbessern, setzen einige Emulatoren entweder eine erhöhte Taktrate ein oder verwenden einen anderen Zeitgeber als den Standardzeitgeber. DummyNet gehört zur ersten Gruppe und NISTNet sowie NetEm zur zweiten.

Leistungsanalyse Wie in anderen bereits vorgestellten Analysen beschrieben, wird auch in [NR09] die mögliche Datenrate zur Messung der Leistungsfähigkeit verwendet. Für die Untersuchungen wird das Messtool iperf mit TCP-Paketen eingesetzt. Die Ergebnisse zeigen, dass bis zu einer geforderten Datenrate von 500 MBit/s alle drei Emulatoren akkurat arbeiten. Erst ab dieser Rate werden Unterschiede deutlich, welche auf die verschiedenen Implementierungen zur

Ratenbegrenzung zurückzuführen sind. DummyNet und NISTNet berechnen die nötige Verzögerung und fügen diese bestimmten Paketen hinzu um die geforderte Datenrate zu erreichen. Die Abweichungen entstehen dabei ebenfalls durch die taktgesteuerte Paketverarbeitung.

NetEm dagegen verwendet dazu Token-Bucket-Algorithmen. Diese erfüllen ursprünglich einen anderen Zweck und gestatten kurzfristige Paketraten, welche über der geforderten Rate liegen können. Für die Emulation hat diese Ausnahmeregelung eher negative Auswirkungen auf die Datenratenbegrenzung.

Fazit Die Arbeit von Nussbaum et al. konzentriert sich überwiegend auf die Implementationen des Zeitgebers sowie der Warteschlangen der drei Emulatoren und zeigt wo aufgrund der jeweiligen Umsetzung bereits im Vorfeld Abweichungen entstehen können.

Im Mittelpunkt der ersten Betrachtungen steht der Grad der Abweichung der Paketverzögerung bei unterschiedlichen Taktungen des Zeitgebers. Anders als bei den Messungen zur Genauigkeit der Paketverzögerung von KauNet (vgl. Abschnitt 3.2.3), werden hier bewusst sehr hohe Paketraten gewählt. Dadurch kann die Implementation des Zeitgebers mit betrachtet werden und es lassen sich, gleichzeitig zur Genauigkeit, auch Aussagen zur Verarbeitungsleistung treffen.

Die Einhaltung der eingestellten Datenrate ist der Schwerpunkt der anderen Betrachtungen. Je nach Implementation, kann im Vorfeld bereits mit einer zu hohen oder zu niedrigen Datenrate gerechnet werden. Beispielsweise sorgen die von NetEm eingesetzten Token-Bucket-Algorithmen für eine zeitweise zu hohe Datenrate.

3.3.2 Shaikh, J. und Minhas, T. et al. (2010)

Die Autoren Shaikh et al. untersuchen in [Sha+10] die drei Emulatoren NISTNet, NetEm und KauNet in Bezug auf ihre „Verzögerungsformung“ (engl. delay shaping). Dabei werden zudem die Abhängigkeiten von der zugrundeliegenden Hardware betrachtet. D. h., die Experimente finden mit annähernd gleichen technischen Voraussetzungen zum einen auf der Hardware des Herstellers AMD (Advance Micro Devices) und zum anderen in einer Intel basierten Umgebung statt. Es zeigt sich, dass die Wahl der Hardware tatsächlich Auswirkung auf die Verzögerungen hat.

Ein weitere Sonderstellung dieser Studie ist der Einsatz des *Coefficient of Throughput Variation* (CoTV). Mit diesen Variationskoeffizienten gelingt es, sich einen einfachen Überblick über die Variation der Datenrate und damit auch der Verzögerung für verschiedene Zeitintervalle zu verschaffen. Dadurch wiederum lassen sich Bewertungen bzgl. der Stabilität emulierter Verzögerungen abgeben. Shaikh et al. beschreiben in [MFA10] ausführlich die Vorzüge dieser Metrik.

Zunächst wird der Versuchsaufbau und anschließend die Vorgehensweise bei der Wertermittlung vorgestellt.

Versuchsaufbau Genutzt wird eine Versuchsumgebung namens „Distributed Passive Measurement Infrastructure“ (DPMI) [Arl05]. Diese besteht aus zwei Messpunkten und der üblichen Verkettung zweier Endpunkte über einen Router, auf dem sich die Emulatoren befinden. Die Endsysteme bestehen für die Betrachtung der Abhängigkeit vom Hardware-System entsprechend entweder aus AMD- oder Intel-basierten Komponenten.

Ein Messpunkt ist für genau eine Richtung der Verbindung zuständig und ist auf eine Weise mit dem Aufbau verbunden, dass dieser den Router überbrückt. Dadurch können Pakete vor und nach dem Router erfasst werden. Die Messpunkte verfügen über „DAG3.5E“[13x] Netzwerkkarten, welche spezielle für die Paketerfassung konzipiert wurden. Eine Veranschaulichung des Versuchsaufbaus befindet sich in Abb. 3.16.

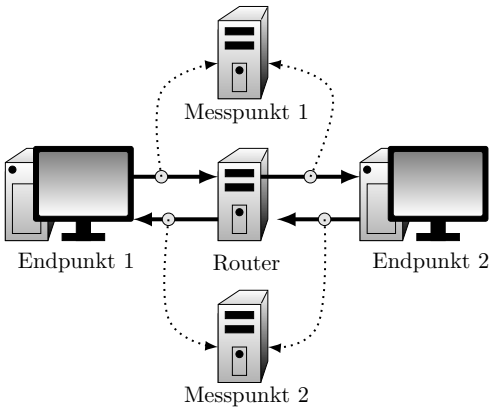


Abbildung 3.16: Versuchsaufbau in [Sha+10] angelehnt an die DPMI.

Gemessen werden das OWD bzw. die Durchgangsverzögerung der einzelnen Pakete. Neben der Hardware werden dabei auch unterschiedliche Paketgrößen verwendet.

Genauigkeitsanalyse Jedem Emulator werden vier verschiedene Verzögerungen fest vorgegeben. Anschließend wird in 32 Versuchsdurchläufen mit 10000 Paketen und unterschiedlichen Hardware-Komponenten die Durchgangsverzögerung jedes dieser Paket erfasst. Anhand dieser Werte werden die durchschnittliche, minimale sowie maximale Verzögerung berechnet. Zur Veranschaulichung werden zusätzlich die verschiedenen CoTV und die Häufigkeitsverteilung je Hardware-Plattform visualisiert. Abbildung 3.17 zeigt ein Beispiel für diese Diagramme bei der Nutzung von KauNet. Zu sehen sind die Häufigkeitsverteilungen für verschiedene Paketgrößen und die Variationskoeffizienten CoTV für die am Emulator ein- sowie ausgehenden Pakete in den Mittlungsintervallen von 1 *ms* bis 1 *s*. Aus den CoTV lässt sich ablesen, wie stark die Paketverzögerungen nach dem Durchlaufen des Emulators je Mittlungsintervall variieren. Je geringer dabei das Intervall ist, desto dominanter treten Spitzenwerte bzw. Ausreißer in Erscheinung. Ist die Verzögerungsausgabe des Emulators instabil entstehen öfter Spitzenwerte und somit verläuft die Kurve für die ausgehenden Pakete steiler. Gleichzeitig zeigt die Kurve der eingehenden Pakete die Grundvariationen des Versuchsaufbaus an. Im Idealfall verursacht der Emulator keine zusätzlichen Variationen und die Kurven decken sich beide.

Insgesamt zeigt sich, dass die Paketverzögerungen auf der Intel-Plattform stabiler sind. Außerdem scheinen die Verzögerungsmechanismen von NetEm am präzisesten zu arbeiten. Starke Schwankungen der emulierten Verzögerungen sind vor allem bei KauNet zu sehen.

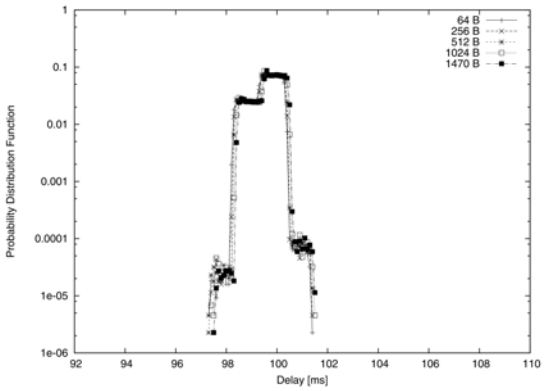


Fig. 15. Fixed Delay KauNet Shaper Intel Platform

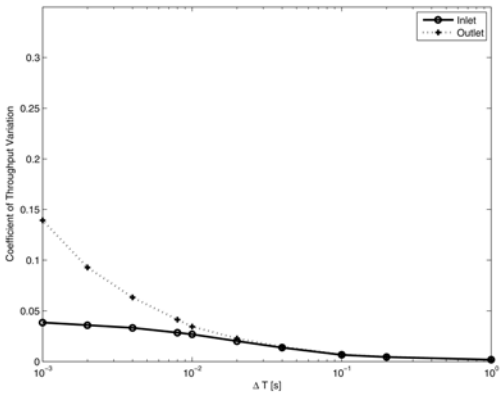


Fig. 16. CoTV for 64 Byte PDUs via the KauNet Shaper on the AMD Platform, nominal delay 100 ms.

Abbildung 3.17: Beispiel für die Visualisierung der Häufigkeitsverteilung und der verschiedenen CoTV. (links: Häufigkeitsverteilung, Quelle: [Sha+10])

Fazit Die Arbeit [Sha+10] konzentriert sich auf die Untersuchungen der Wertstabilität einer vorgegebenen Paketverzögerung. Erstmals wurde hierbei auf die zugrundeliegende Hardware-Umgebung eingegangen und es zeigten sich Abhängigkeiten diesbezüglich.

Interessant ist ebenfalls die Versuchsumgebung, welche auf DPMI aufbaut und es ermöglicht passive Messungen am Ein- und Ausgang des Emulators durchzuführen ohne Paketerfassungswerkzeuge auf diesem zu installieren. Die Messungen am Ein- und Ausgang gestatten es, die tatsächliche Verzögerung des Emulators getrennt von der Grundverzögerung des Versuchsaufbaus zu betrachten. Eine Uhrensynchronisation entfällt ebenfalls.

Eine weitere Neuerung im Vergleich zu den anderen vorgestellten Veröffentlichungen ist die Verwendung der CoTV-Metrik, um so einen Überblick über die Höhe der Variation für verschiedene Mittlungsintervalle zu erhalten.

3.3.3 Minhas, T. und Shaikh, J. et al. (2011)

Auch diese Studie stammt von den Autoren Minhas et al. und untersucht den Einfluss verschiedener Hardware-Umgebungen und Paketgrößen auf das Emulationsverhalten. Jedoch steht in [Min+11] die Untersuchung der Genauigkeit bei der Datenratenbegrenzung durch NetEm und KauNet im Vordergrund. Zusätzlich werden mit Hilfe der gewonnenen Messwerte Rückschlüsse auf die Hauptfaktoren bei der Datenratengestaltung des jeweiligen Emulators gezogen. Diese Hauptfaktoren sind zum einen die Puffergröße und zum anderen die Paketrate.

Versuchsaufbau In [Min+11] wird der gleiche Aufbau verwendet wie in der zuvor vorgestellten Arbeit [Sha+10] (siehe Abb. 3.16).

In diesem Fall wird die ein- sowie ausgehende Datenrate am Emulator ebenfalls für verschiedene Paketgrößen und Hardware-Komponenten gemessen. Die Emulatoren beschränken je nach Versuchsdurchgang die Datenrate auf 512 KBit/s oder 4000 KBit/s.

Genauigkeitsanalyse Berechnet werden das Verhältnis S_R aus der ausgehenden Datenrate R_o und der vorgegebenen Datenrate R_s. Ist der Wert dieses Verhältnisses gleich eins, so arbeitet der Emulator ideal. Dagegen sind Werte mit $S_R << 1$ ein Hinweis darauf, dass entweder für $R_s < R_i$ der Emulator überlastet ist oder für $R_s > R_i$ keine Ratenbegrenzung notwendig ist. Es zeigt sich, dass die vorgegebene Datenrate für große Paket eher umgesetzt wird als für Pakete mit einer geringen Größe. Allerdings scheint NetEm dennoch weniger Probleme mit einer kleinen Paketgröße zu haben als KauNet. Zur genaueren Analyse sowie zur Visualisierung werden zusätzlich die relativen Abweichungen $R_o/R_s - 1$ als Diagramm dargestellt und neben der durchschnittlichen Paketrate P_o auch die Puffergrößen X_C der Emulatoren wie folgt berechnet:

$$P_o = \frac{R_o}{L} \tag{3.2}$$

$$X_C = \frac{D \cdot R_o}{L} \tag{3.3}$$

Wobei D der Paketverzögerung und L der Paketgröße entsprechen. Zu beachten ist, dass D für den Überlastfall des Emulators gemessen wird. Die Ergebnisse zeigen, dass NetEm 100 und KauNet 50 Pakete zwischenspeichern kann.

Weiterhin zeigt sich für KauNet, das dieser unabhängig von der zugrundeliegenden Hardware arbeitet. Wogegen NetEm in der AMD-basierten Umgebung Datenraten akkurater umsetzt.

Fazit Der Einsatz von DPMI als Versuchsumgebung, gestattet auch in dieser Arbeit eine präzise Untersuchung des ein- und ausgehenden Verkehrs. Interessant ist vor allem die Möglichkeit, indirekt über die Messung der ausgehenden Datenrate und Paketverzögerung auf die Puffergröße des Emulators schließen zu können. Dieser Puffer stellt u. a. für die Leistungsbetrachung eines Emulators ein wichtiges Merkmal dar.

3.4 Zusammenfassung

Aus den Angaben in den vorgestellten Publikationen ergeben sich die in Tabelle 3.10 zusammengefassten feststehenden Eigenschaften des jeweiligen Emulators. Aufgrund der Bedeutung des Zeitgebers wurde dieser mit in die Tabelle aufgenommen. Ebenso zeigten die Recherchen, dass viele Emulatoren neben Benutzeroberflächen auch Wahrscheinlichkeitsverteilungen für Paketeffekte sowie Verzögerungen anbieten, was für die Generierung von Pfadangaben und einer gewissen Dynamik von Vorteil sein kann. Daher werden auch diese Punkte mit in die Taxonomie aufgenommen.

Keiner der vorgestellten Emulatoren deckt alle Eigenschaften in der Taxonomie ab. In Bezug auf die Mobilfunkeffekte bietet einzig WANem Unterbrechungen an. Ausbreitungsmodelle oder Handover wurden bei keinem Emulator gefunden. Der von den Effekten her umfangreichste Emulator ist der „Linktropy 7500 pro“. Dieser verfügt über alle Paketeffekt und kann wie WANem, NetEM oder Imunes über eine Benutzeroberfläche konfiguriert werden. Letzterer hat eine indirekte Pfadvorgabe und die Nutzung von virtuellen Betriebsmitteln als Alleinstellungsmerkmale. Alle anderen setzen die Vorgaben explizit um und gehören zur Gruppe der NLE. Imunes kann durch eine Erweiterung auf einer dezentralen Architektur aufsetzen. ModelNet dagegen beruht

von vorn herein auf einer dezentralen Architektur. Alle übrigen werden zentral betrieben. Der Arbeitsbereich im Kernel ist wiederum allen gemein.

Weiterhin hat sich herausgestellt, dass die Produktangaben zu den meisten Emulatoren deutliche Lücken aufweisen. So ist der Umfang dieser Angaben z. B. bei den Emulatoren DummyNet (Tab. 3.3) und ModelNet (Tab. 3.5) im Vergleich zu den Produktangaben zum Hardware-Emulator in Tabelle 3.1 sehr gering. Zumeist fehlen die Arbeitsbereiche und die mögliche Granularität für die verschiedenen Effektqualitäten. Daher werden an dieser Stelle keine Vergleiche zwischen den Emulatoren angestellt.

Auffällig ist ebenso, dass in den veröffentlichten Analysen hauptsächlich die Genauigkeit der emulierten Paketverzögerung und Datenrate betrachtet wird. Untersuchung zu den übrigen *Effektqualitäten* wurden, vermutlich aus Gründen des Umfanges, nicht veröffentlicht. An den genannten Stellen setzen daher eigene Untersuchungen an, um so die fehlenden Angaben zu ermitteln.

Dennoch lassen sich viele Hinweise für eigene Untersuchungen herausfiltern. Dazu gehören Erkenntnisse über den Versuchsaufbau, Metriken und Visualisierungsmöglichkeiten sowie Betrachtungen, welche bereits im Vorfeld einer Messung auf Abweichungen hindeuten. In den Arbeiten zu DummyNet werden die Einflussfaktoren CT, Prozessinterferenzen und die Taktrate untersucht. Außerdem wird die Metrik PPT als Alternative zur PPS vorgestellt. Wobei jedoch die direkte Implementierung zur Messung der PPT außen vor gelassen wird. Die Publikationen zu KauNet beschreiben einen Versuchsaufbau bei dem mit passiven Messmethoden direkt am Ein- und Ausgang des Emulators sowie an den Endpunkten Pakete erfasst werden. Dadurch können die Durchgangswerte des Emulators von den Grundwerten des Versuchsaufbaus getrennt werden. Für die Genauigkeit der emulierten Verzögerung werden anders als bei der Studie von Nussbaum et al. kleine Pakete mit einer geringen Paketrate verwendet, um so Systemeinflüsse zu minimieren. Nussbaum et al. dagegen verwenden hohe Paketfrequenzen, um die Auswirkungen des Zeitgebertaktes auf die Zuverlässigkeit der emulierten Werte zu untersuchen. Bei ModelNet wird der Begriff der Kapazität eines Emulators eingeführt. Diese zeigt den Punkt im Emulationsprozess an, bei der die CPU-Last 100 % beträgt und unerwünschte Paketverluste auftreten. In der Arbeit zu NetPath wird diese Kapazität auf die Paketrate bezogen und als MLFFR bezeichnet. Für die Evaluierung von NetEm wird ein in einer Realumgebung zuvor aufgezeichneter TCP-Strom nachgebildet. Für die dezentralen Lösungen Imunes und ModelNet werden die Paketraten für steigende Verbundteilnehmer und Netzwerkelemente untersucht. Die Betrachtung der Verzögerung wird in allen Publikationen sowie Studien mit Hilfe von variierenden Paketgrößen und -raten durchgeführt. Für eine übersichtliche Visualisierung der Messergebnisse werden zumeist Häufigkeitsverteilungen und speziell in der Arbeit von Shaikh et al. der CoTV verwendet.

In der folgenden Anforderungsanalyse werden die aus den vorgestellten Analysen gewonnen Erkenntnisse weiter ausgeführt, um so ein eigenes Konzept zur Wertgewinnung zu erhalten.

				LinkTropy	DummyNet	KauNet	ModelNet	NetEm	WANem	Imunes	NetPath
Funktionsweise	Architektur		zentral / dezentral	+ /	+ /	+ /	/ +	+ /	+ /	+ /	+ /
	Arbeitsbereich		Kernel / Nutzer	+ /	+ /	+ /	+ /	+ /	+ /	+ /	+ /
	Zeitgeber		Type / Taktung	? / ?	Modi. / 10 KHz	Modi. / 10 KHz	Modi. / 10 KHz	adapt. / ?	? / ?	? / ?	Polling
	Bedarf	Betriebsystem	Linux / FreeBSD	Hardware	3.8 / 8.x	2.6 / 7.3	2.4 / 4.11	3.8 /	3.8 /	/ 8.3	2.4 /
		Mindest-anforderung	notw. Rechnern	0	1	1	2	1	1	1	1
			CPU[GHz] / RAM[GB]		2.4 / k.A.	2 / 4	1.4 / 1	1.4 / 1	1.4 / 0.5	/	/
	Pfadvorgabe	explizit	statisch / dynamisch	+ /	+ /	+ / +	+ /	+ /	+ /		+ /
		implizit	konstant / variabel							+ / +	
	Benutzeroberfläche			+				+	+	+	
Effektqualität	Nach-bildung	probabilistisch	Verteilungsarten	uni./norm.	uni.		uni.	uni./norm./pare.	norm./pare.		
		deterministisch		Konst.		Muster					
	Daten-rate	maximale Datenrate C		+	+	+	+		+		+
		Cross-Traffic CT		+							
	Verzö-gerung	Einwege OWD		+	+	+	+	+	+	+	+
		Variation δOWD		+	+	+	+	+	+		+
	Pakeeffekte	Verlust		+	+	+	+	+	+		+
		Bitfehler		+		+		+	+	+	+
		Dopplung		+				+	+	+	+
		Neuordnung		+	+	+	+	+	+	+	+
	Mobilfunk	Ausbreitungs-modell	empir. / semi. / determ.								
		Handover	hard / soft								
		Unter-brechungen	Idle / Abbr. / Ab-Aufb.						+/+/+		
Skalier-barkeit	Netzknotenemulation			+	+	+	Core +	+	+		+
	Emulation virtueller Netzwerke		VM / VB				VNs /			+	

Tabelle 3.10: Übersicht der feststehende Eigenschaften der ausgewählten Emulatoren.

4 Anforderungsanalyse

Bevor mit der Konzeption für ein Messverfahren und dessen Implementation in Form einer Analyse-Software begonnen werden kann, muss erst dessen Rahmen mit Hilfe von funktionalen und qualitativen Anforderungen sowie Abgrenzungskriterien abgesteckt werden. Diese lassen sich zum einen aus der Aufgabenstellung entnehmen und zum anderen aus den Erläuterungen zu den vorgestellten Analysen in Kapitel 3 gewinnen. Letztere bieten eine Vielzahl an Hinweisen zum *Versuchsaufbau*, zu *Vorfeldbetrachtungen*, zur *Versuchsdurchführung* sowie zur *Auswertung* der Messwerte und zeigen dadurch viele Anforderungen an ein eigenes Messkonzept auf. Die genaue Aufschlüsselung der Anforderungen geschieht in den folgenden beiden Abschnitten und wird im letzten Abschnitt tabellarisch dargestellt.

4.1 Funktionale Anforderungen

Die Aufgabenstellung beinhaltet die grundlegenden Anforderungen an das Ergebnis dieser Arbeit. Im wesentlichen umfassen diese die Forderung nach:

1. einem ***Verfahren*** für
2. ***Genauigkeitsmessungen*** der in Abschnitt 2.4 erwähnten Metriken für die Datenrate, Verzögerung sowie die Paketeffekte,
3. bei gleichzeitigen ***Leistungsmessungen*** für
4. möglichst viele Hard-und Software basierte Emulatoren
5. und dessen ***Implementation*** in Form einer Software,
6. welche in der Lage ist, Messergebnisse für eine anschließende ***Auswertung*** zu liefern.

Für die Bestimmung der funktionalen Anforderungen an das Verfahren sowie dessen Umsetzung werden diese verallgemeinert Forderungen anhand der gewonnen Erkenntnisse weiter konkretisiert.

4.1.1 Verfahren

Das Messkonzept bzw. Messverfahren ist eines der angestrebten Ziele dieser Arbeit und beinhaltet neben der konkreten Vorgehensweise auch den Versuchsaufbau.

Vorgehensweise In den vorgestellten Studien hat es sich als vorteilhaft erwiesen, bereits vor der eigentlichen Analyse, Untersuchungen bzgl. des Taktgebers (FV.1, vgl. Tab 4.3) und der Puffergröße (FV.2, vgl. Tab 4.3) des Emulators durchzuführen. Gleichzeitig sollten die Grundverzögerungen (FV.3) des Versuchsaufbaus mit betrachtet werden. Somit sollte auch im eigenen Messkonzept eine Vorfelduntersuchung eingeplant werden. Ebenso sollte im Anschluss einer Messung neben der direkten Ausgabe zusätzlich eine Auswertung der Messwerte erfolgen. Dazu zählt beispielsweise eine Häufigkeitsverteilung (FA.1) bei Verzögerungswerten. Abbildung 4.1 fasst die geforderte Vorgehensweise anhand der genannten Punkte zusammen.

Abbildung 4.1: Grundlegende Vorgehensweise

Versuchsaufbau Weiterhin geht aus den Erkenntnissen in Kapitel 3 hervor, dass die Versuchsumgebung neben dem Emulator eine Paketquelle beinhalten muss. Diese Quelle sollte dabei in der Lage sein, Probepakete mit definierten Eigenschaften zu generieren, um u. a. Paketströme mit unterschiedlichen Raten nachbilden zu können. Der Paketgenerator (FT.1, vgl. Tab 4.3) sollte daher Einstellungsmöglichkeiten bzgl. der Paketgröße (FI.5), Protokollart (FI.6) und des zeitlichen Paketabstandes (FI.7) anbieten.

Entsprechend zur Quelle, müssen Messpunkte für die Erfassung der Pakete (FT.2, vgl. Tab 4.3) vorhanden sein. Da der Versuchsaufbau möglichst unabhängig von einer bestimmten Hardware sein soll, werden die Pakete mittels Software erfasst. D. h. anstelle von einer beispielsweise „DAG3.5E“-Netzwerkkarte, werden die Pakete direkt von der Empfängerapplikation oder mittels spezieller Treiber aufgezeichnet. Zudem werden für die Paketerfassung Filter (FT.5) sowie eine Datenhaltung (FT.6) benötigt, um anschließende Analysen und Berechnungen nach verschiedenen Gesichtspunkten zu ermöglichen. Speziell für die Untersuchungen der OWD ist außerdem eine Zeitsynchronisation (FT.4) zwischen den Messpunkten des Versuchsaufbaus von Nöten.

Für die Ermittlung der emulierten Datenraten, Verzögerungen und Paketeffekte wird ein Tool benötigt, welches diesbezügliche zeit- und ereignisgesteuerte Analysen sowie Berechnungen mit den erfassten Paketen anstellen kann (FT.3). Die genannten Forderungen werden im Schema 4.2 zusammengefasst und in Beziehung gebracht.

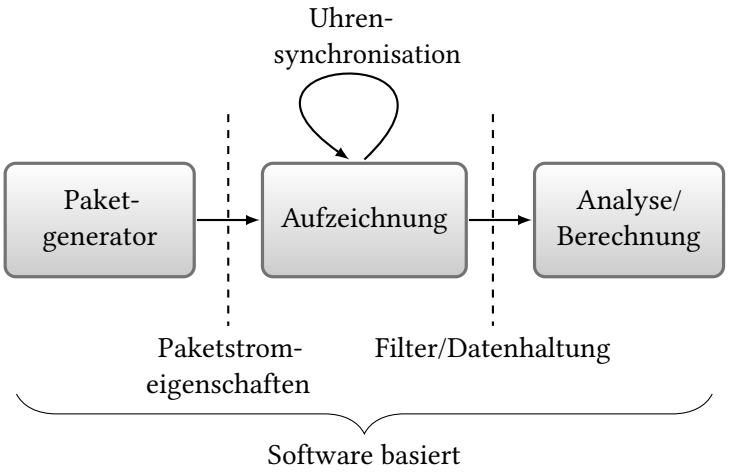


Abbildung 4.2: Grundlegende Anforderungen an den Versuchsaufbau

4.1.2 Genauigkeits- und Leistungsmessungen

Im Mittelpunkt der Analysen zu einem Emulator stehen die in den Abschnitten 2.4.1 bis 2.4.3 erwähnten Datenraten, Verzögerungen und Paketeffekte bei unterschiedlichen Voraussetzungen wie z. B. der Pfadlänge. Die genaue Aufzählung der geforderten Netzwerkeffekte befindet sich in Tabelle 4.3. Die weiteren Anforderungen betreffen die Verringerung möglicher *Abweichungen* und die *Zielstellung* der Messungen.

Abweichung Während der Analysen zur Datenrate und Verzögerung dürfen keine ungewollten Paketverluste auftreten (FM.1), da sonst Messwerte verfälscht werden könnten. Zudem zeigte sich in den vorgestellten Publikationen, dass gleichzeitig zur eigentlichen Messung die Datenrate des Paketgenerators (FM.2) und die durch diesen verursachte Verzögerung mit betrachtet werden sollten, um mögliche Abweichungen zu berücksichtigen.

Bei Werterfassungen innerhalb des Emulatorsystems dürfen die Emulationsprozesse nicht beeinträchtigt werden (FM.3).

Eine weitere Erkenntnis aus dem vorangegangen Kapitel ist, dass reale Testumgebungen gegenüber hybriden oder virtuellen die genauesten Ergebnisse liefern. Daher sollen auch die eigenen Untersuchungen innerhalb einer realen Umgebung stattfinden (FT.7).

Zielstellung Die Endergebnisse der Messungen sollen den möglichen Arbeitsbereich sowie die Wertstabilität des Emulators in Verbindung zu dessen Auslastung und bei einer vorgegebenen Toleranz bestimmen (FI.8). Dazu sind die Effekte zunächst einzeln und anschließend gemeinsam,

```
Eingabe:
--------
+--------------------+---------+...
| Pfadabschnitte     |         |
| seriell | parallel | Toleranz|
+---------+----------+---------+...
| 3       | 0        | 0.001 % |
.         .          .         .
.
.

Ausgabe:
--------
+---------+-----------------+---------in-Bezug-auf-Maximalwert----------------------+...
|         | Arbeitsbereich | Stabilitaet  | Abweichung Erwartungswert |     Last    |
| Effekt  | min    | max    | 10 Versuchen | MAD         | relative     | CPU  | RAM |
+---------+--------+--------+--------------+-------------+--------------+------+-----+...
| PLR     | 1 %    | 10 %   | 90 %         | 0.03 %      | 0.01 %       | 1 %  | 3 % |
.         .        .        .              .             .              .      .     .
.
.
```

Abbildung 4.3: Ein- und Ausgabemöglichkeit der geforderten Endergebnisse.

beispielsweise in Form einer nachgestellten Verbindung, zu betrachten (FM.5 und FM.6).

Zur Untersuchung der Skalierbarkeit bzw. Leistung werden je nach Netzwerkeffekt unterschiedliche Paketgrößen, Paketraten, Anzahlen an seriellen und parallelen Pfadabschnitten verwendet. Für letztere bietet es sich an, Erwartungswerte zu berechnen (FM.4), um Referenzpunkte für die Betrachtung der Genauigkeit zu erhalten. Ein Beispiel für eine mögliche Programmeingabe und -ausgabe der Endergebnisse ist in Abb. 4.3 zu sehen.

4.1.3 Implementation

Die praktische Umsetzung des Messkonzepts als Software ist ein weiteres Hauptziel dieser Arbeit. Mit dieser sollen die Analysen zur Genauigkeit sowie Leistung durchgeführt werden und eine Auswertung der Ergebnisse möglich sein. Zusätzlich zu diesen grundlegenden Forderungen bzw. *Musskriterien* bestehen weitere *Wunschkriterien*, welche die Handhabung und Ausgabe erweitern könnten.

Musskriterien Neben der Eingabe der zu verwendenden Metrik, muss das Programm bestimmte Einstellungsmöglichkeiten bieten. Diese lassen sich indirekt aus den Beschreibungen zur Vorgehensweise bei den Messungen in den vorgestellten Studien und Publikationen ablesen. Aus diesen geht hervor, dass Vorgaben bzgl. der Generierung und Erfassung von Paketen sowie der Verbindung und Zeit notwendig sind. Dadurch soll es beispielsweise möglich sein eine definierte Paketrate über vorgegebene Netzwerkschnittstellen bzw. Netzwerkports zu versenden und nach bestimmten Zeitabständen zu erhöhen. Die genaue Auflistung der geforderten Einstellungsmöglichkeiten ist in Tabelle 4.3 FI.1 bis FI.8 zu sehen.

Wunschkriterien Mit Hilfe der Software könnte eine vollautomatische Analyse des Emulators durchgeführt werden. D. h. der Nutzer tätigt die Eingaben und alle Einstellungen am Emulator sowie die Messungen für einen bestimmten Betrachtungsbereich werden automatisch durchgeführt (FI.w1).

Zur einfacheren Steuerung der Implementation könnte eine GUI eingesetzt werden (FI.w2). Diese wiederum könnte die Endergebnisse als Diagramme visualisieren (FI.w3). Wobei auch Ergebnisse verschiedener Emulatoren gleichzeitig betrachtet werden könnten (FI.w4).

4.1.4 Auswertung

Um eine einfache Vergleichsmöglichkeit zwischen den verschiedenen Emulatoren zu schaffen, sollten die absoluten Messwertangaben mit statischen Hilfsmitteln aufbereitet werden.

In den Publikationen und Studien wurden zumeist Häufigkeitsverteilungen (FA.1) verwendet, um Interpretationen über die Genauigkeit bei bestimmten Vorgaben, wie z. B. einer Paketlänge, zu ermöglichen. Daher sollte neben der den Angaben zum Minimal, Maximal und Durchschnittswert auch die Verteilung der Werte ausgegeben werden (FA.3 bis FA.5). Für die Analyse der Abweichung bietet sich außerdem die Ausgabe der mittleren absoluten Abweichung *(engl. mean absolute deviation (MAD))* sowie die des relativen Fehlers an (FA.6 und FA.7).

Um die Stabilität bzw. die Variation einer emulierten Datenrate zu betrachtet, sollte der vorgestellte CoTV verwendet werden (FA.2).

Für die Auswertung der Voruntersuchungen werden Algorithmen benötigt, welche aus den Messwerten die Taktrate und die Puffergröße ermitteln.

4.2 Qualitative Anforderungen

Die qualitativen Anforderungen fassen zum einen die erforderlichen *Nachweise* zur korrekten Funktion des Messverfahren bzw. dessen Implementation zusammen und zum anderen beinhalten diese alle *nicht-funktionalen Anforderungen*.

Teilziel	Bezeichnung
Vorfeldanalyse	FV
Messkonzepte	FM
Auswertung	FA
Testaufbau	FT
Implementation	FI
nicht-funktionale Anforderungen	QF
Evaluation/Nachweis der Umsetzung	QN
Messungen	M

Tabelle 4.1: Übersicht Teilziele

Nachweise Zunächst muss der korrekte Routenverlauf der Probepaket nachgewiesen werden, um zu zeigen, dass alle Pakete direkt vom Sender über den Emulator zum Empfänger gesendet werden (QN.1). Dadurch soll sichergestellt werden, dass nur der Emulator an den verschiedenen Netzwerkeinflüssen beteiligt ist.

Als nächstes muss die richtige Funktion des Paketgenerators gezeigt werden (QN.2). Einstellungen bzgl. der Paketgröße und -rate müssen weitestgehend genau und stabil umgesetzt werden, um exakte Ergebnisse zu erhalten. Daher sollten mindestens die verwendeten Paketraten und -größen am Generator getestet und Abweichungen dokumentiert werden.

Weiterhin sollte die Paketerfassung überprüft und mit unabhängigen Programmen wie Wireshark verglichen werden (QN.3). Nur so kann sichergestellt werden, dass die Messdaten korrekt aufgezeichnet werden.

Eventuell auftretende Fehlkonfigurationen sollten vom Messsystem gemeldet werden (QN.4). Für den Nachweis einer Fehlerbehandlung sollte daher eine falsche Konfiguration bzgl. der Netzwerkeinstellung und Messparameter gemeldet werden.

nicht-funktionale Anforderungen Die Installation des Programms soll mit nur geringem Aufwand erfolgen. D. h. alle benötigten Programm-Bibliotheken müssen frei verfügbar sein oder mitgeliefert werden.

Der Versuchsaufbau muss weitestgehend unabhängig von spezieller Hardware sein. Daher dürfen keine besonderen Netzwerkkarten oder Router verwendet werden. Außerdem dürfen Änderungen von Messparametern oder zu untersuchenden Netzwerkeffekten keine Änderungen am Versuchsaufbau mit sich ziehen.

Bei der Verwendung von fertigen Tools, sollen diese in den Programmablauf integriert werden. Das bedeutet, dass der Nutzer nur einmal alle Eingaben tätigt und auch nur ein Programm startet. Alle anderen Konfigurationen übernimmt die geforderte Implementation.

Der fertigen Implementation muss eine Dokumentation beliegen, welche die Handhabung des Programmes und dessen Ausgaben erklärt.

	qualitative Anforderung	Bezeichnung
Nachweise	Routenverlauf Sender-Emulator-Empfänger	QN.1
	genaue und stabile Paketgenerierung	QN.2
	korrekte Paketerfassung	QN.3
	Behandlung v. Fehlkonfigurationen	QN.4
nicht-funktional	einfache Installation, freie Bibliotheken	QF.1
	Hardwareunabhängig	QF.2
	gleichbleibender Testaufbau	QF.3
	Dokumentation	QF.4
	Integration vorhandener Tools	QF.5

Tabelle 4.2: Übersicht qualitative Anforderungen

4.3 Abgrenzungskriterien

Einige der in Abschnitt 2.4 erwähnten *Effektqualitäten* werden von der Umsetzung nicht abgedeckt. Dazu gehören die Mobilfunkeffekte und die Bitfehlerrate. Nicht korrigierbare Bitfehler äußern sich auf Paketebene als Verlust. Daher sind diese Fehler nur indirekt über die PLR zu ermitteln. Von den untersuchten Emulatoren beherrscht lediglich WANem Mobilfunkeffekte. Die von WANem unterstützten Effekte beschränken sich zudem auf Unterbrechungen. Daher wird auf die Untersuchung der Mobilfunkeffekte verzichtet.

Weiterhin können im Rahmen dieser Arbeit nicht alle vorgestellten Emulatoren analysiert werden. Dazu zählen die Emulatoren ModelNet und NetPath. Beide setzten Betriebssysteme voraus, welche bereits mehrere Jahre alt sind und entsprechend alte Hardware fordern, die wiederum nicht zur Verfügung stehen.

Ebenso können nicht alle Ergebnisse für jeden Versuch vorgestellt werden, da die Anzahl der Einzelmessung bei über 92000 liegt. Daher werden nur einige von diesen präsentiert.

4.4 Zusammenfassung

Aus der Analyse der Aufgabenstellung und unter der Berücksichtigung der gewonnenen Erkenntnisse aus Kapitel 3 ergeben sich die in Tabelle 4.1 aufgelisteten Teilziele. Diese werden, bis auf das letzte Teilziel, nach funktionalen (F) und qualitativen (Q) Anforderungen gruppiert. Das letzte Teilziel steht gesondert, da es die praktische Anwendung betrifft und die Endergebnisse dieser Arbeit liefert. Die konkreten funktionalen und qualitativen Anforderungen sind in den Tabellen 4.3 und 4.2 mit der verwendeten Aufteilung dargestellt. Die Buchstabenkombination in der Spalte „Bezeichnung“ deutet dabei das übergeordnete Teilziel an.

		funktionale Anforderung	Bezeichnung
Messverfahren	Testaufbau	Paketgenerator	FT.1
		Paketerfassung	FT.2
		Analyse/Berechnung	FT.3
		Uhrensynchronisation	FT.4
		Filter	FT.5
		Datenhaltung	FT.6
		Reale Umgebung	FT.7
	Messungen	keine ungew. Paketverluste	FM.1
		Kontrolle des Paketgenerators	FM.2
		keine Beeintr. des Emulators	FM.3
		Berechnung Erwartungswerte	FM.4
		einzelne Netzwerkeffekte	FM.5
		mehrere Netzwerkeffekte	FM.6
		Metriken (FM.7.1 bis FM.7.7)	FM.7
	Auswertung	Häufigkeitsverteilung	FA.1
		CoTV	FA.2
		Minimum	FA.3
		Maximum	FA.4
		Durchschnitt	FA.5
		MAD	FA.6
		relative Abweichung	FA.7

Tabelle 4.3a: Messverfahren

		funktionale Anforderung	Bezeichnung
Metriken	Vorfeld	Taktrate	FV.1
		Puffergröße	FV.2
		Grundverzögerung	FV.3
	Messungen	verfügbare Datenrate	FM.7.1
		Massendurchsatzrate	FM.7.2
		Verlustrate	FM.7.3
		Dopplungsrate	FM.7.4
		Neuordnungsrate	FM.7.5
		Verzögerung	FM.7.6
		Verzögerungsvarianz	FM.7.7
Einstellungsmöglichkeiten	Musskriterien	Auswahl der Metriken	FI.1
		Zeit: Messdauer, Intervall	FI.2
		Messart: Einwege, Umlauf	FI.3
		Verbindung: Ports	FI.4
		Paketgröße	FI.5
		Protokoll: UDP	FI.6
		Datenrate/Paketrate	FI.7
		Toleranz	FI.8
	Wunschkrit.	vollautom. Analysen	FI.w1
		GUI	FI.w2
		Diagramme	FI.w3
		Vergleich Emulatoren	FI.w4

Tabelle 4.3b: Metriken und Implementation

Tabelle 4.3: Übersicht funktionale Anforderungen

5 Konzeption

Bevor auf die Vorgehensweisen für die Vorfelduntersuchungen und Messungen eingegangen wird, werden zunächst die Komponenten des Versuchsaufbaus aus Hardware- sowie Softwaretechnischer Sicht erläutert. Für erstere wird dabei, basierend auf den vorgestellten Arbeiten, eine eigene Lösung schrittweise konstruiert.

Bis auf die OWD, können die Konzepte zur Ermittlung der übrigen Metrikwerte aus der Arbeit [Büs13] zum Analysewerkzeug Nora mit geringen Anpassungen übernommen werden. Das fehlende Konzept und die Anpassungen sind Teil des letzten Abschnittes dieses Kapitels. Zudem werden die Berechnungen der statistischen Werte FA.1 bis FA.6 im letzten Abschnitt vorgestellt. Wobei neben dem CoTV als Grundlage eine eigene Statistik namens „Variationskoeffizient der Verzögerungen mit blockierenden gleitenden Mittelwerten“ (engl. *Coefficient of Delay Variation with Blocking Moving Average* (CoDVwBMA)) eingeführt wird.

5.1 Versuchsaufbau

Der richtige Versuchsaufbau ist Grundvoraussetzung für die Messungen. Dessen Eigenschaften entscheiden über die möglichen Untersuchungen und deren Messgenauigkeit. Daher werden in diesem Abschnitt die in Kapitel 3 vorgestellten Testaufbauten diskutiert und einer eigenen Variante gegenüber gestellt. Weiterhin gehören die in der Anforderungsanalyse (vgl. Abb. 4.2) genannten softwareseitigen Komponenten und Messwerkzeuge mit zur Versuchsumgebung. Auch für diese werden bereits vorhandene Lösungen abgewägt und eine eigene Konzeption vorgestellt.

5.1.1 Hardwaretechnischer Aufbau

Im Kapitel 3 wurden u. a. von Nussbaum et al., Shaikh et al. und Hall et al. unterschiedliche Versuchsumgebungen vorgestellt, die zusammen die Basis für einen eigenen Ansatz bilden. Die Hauptbestandteile sind dabei ein Paketsender sowie -empfänger, der Emulator und Messpunkte. Die Vernetzung der Bestandteile und die Platzierung der Messpunkte sind die Unterscheidungsmerkmale zwischen den drei Aufbauten.

Der Aufbau, welcher von Nussbaum et al. verwendet wird (siehe Abb. 3.14), ist Teil des französischen Grid’5000-Projektes [13g] und besteht im wesentlichen aus einer Sterntopologie mit einem Router als Knotenpunkt. Der Sender und der Empfänger sowie der Emulator bilden die Endpunkte. Das Projekt ist für großflächige Forschungen im Bereich der verteilten Anwendungen entworfen worden und setzt sich zum Teil aus Cloud-basierten VMs zusammen. Die Vorteile eines solchen Systems sind die hohe Verfügbarkeit und Erweiterbarkeit. Für die Untersuchung der Genauigkeit und Leistung eines einzelnen Emulators sind jedoch die genannten Vorteile hintergründig. Vielmehr sollten so wenige Netzwerkkomponenten wie möglich an der Versuchsdurchführung beteiligt sein, um die Anzahl der ungewollten Paketeinflüsse zu minimieren. Daher wird für den

eigenen Aufbau auf den Router verzichtet und die in Abb. 3.14 gezeigte logische Verbindung, physisch umgesetzt. Der Emulator befindet sich dadurch direkt zwischen dem Sender und dem Empfänger. Dafür kann der Emulator entweder mit zwei einzelnen Netzwerkkarten oder einer Dual-Port-Netzwerkkarte ausgerüstet werden. Der erste Ansatz der eigenen Versuchsumgebung ist in Abbildung 5.1 dargestellt.

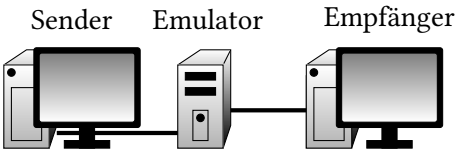


Abbildung 5.1: Erster Ansatz des eigenen Versuchsaufbaus.

Der nächste wichtige Punkt bei der Planung der Versuchsanlage beinhaltet die Anzahl und die Positionierung der Messpunkte.

Der Versuchsaufbau in [Sha+10] basiert auf der DPMI und positioniert die beiden Messpunkte parallel zum Emulator (vgl. Abb. 3.16). Diese zeichnen passiv den Datenverkehr zwischen den Endpunkten und dem Emulator auf. Die passive Aufzeichnung ermöglicht es den tatsächlichen Datenverkehr inklusive präzisem Zeitstempel für den Versende- und Empfangszeitpunkt für jedes einzelne Paket zu erhalten. Dieser Aufbau stellt daher das Optimum bzgl. der Werterfassung dar, erfordert jedoch spezielle „DAG“-Netzwerkkarten. Eine prinzipielle Nachbildung dieser Struktur ist auch mit aktiven Netzwerkkomponenten möglich. Beispielsweise könnte ein Switch mit „Port-Mirroring“ eingesetzt werden, wodurch anders als beim Hub die verfügbare Datenrate nicht verringert wird. Allerdings werden je nach verwendeter Weiterleitungsmethode zusätzliche Latenzen im Mikrosekundenbereich hinzugefügt. Damit der Switch nicht zum Flaschenhals wird, muss dieser mindestens 1 $GBit/s$ pro Port unterstützen.

Da jedoch ein solcher Gigabit-Switch mit „Port-Mirroring“ nicht zur Verfügung steht, muss eine andere Möglichkeit der passiven Paketaufzeichnung gefunden werden. In [Hal+12] verwenden die Autoren bei der Untersuchung von KauNet dafür das Werkzeug Tcpdump an insgesamt vier Messpunkten. Zwei von diesen befinden sich direkt auf dem Rechner des Emulators. Die anderen beiden sind jeweils dem Sender und dem Empfänger vorgeschalten. Der Vorteil dieser Aufteilung ist, dass unerwünscht auftretende Paketeffekte jeweils zwischen dem Sender und Emulator sowie zwischen dem Empfänger und Emulator erkannt werden können.

Für die Untersuchung in virtuellen oder hybriden Umgebungen ist diese Vorgehensweise durchaus sinnvoll. Der eigene Aufbau wird jedoch mit realen Verbindungen in Form von kurzen Netzwerkkabeln ($\sim 50\ cm$ Länge) konzipiert. Die Einflüsse der Kabelverbindungen auf die Probepakete können vernachlässigt werden, da die Wahrscheinlichkeiten für ungewollte Paketverluste, -dopplungen oder -neuordnungen äußerst gering sind und sich Verzögerungen bei solchen Kabellängen im Nanosekundenbereich befinden. Dadurch wiederum kann auf die zwei Messpunkte des Emulators verzichtet werden. Dieser Verzicht hat auch den Vorteil, dass dem Emulator ohne diese Messpunkte mehr Ressourcen zur Verfügung stehen und keine zusätzlichen Beeinflussungen entstehen.

Der angepasste Ansatz ist in Abbildung 5.2 dargestellt.

Ein weiterer Punkt, welcher von keiner der bisher vorgestellten Versuchsumgebungen berücksichtigt wird, ist die Uhrensynchronisation bei der Messung von OWDs. Trotz eines sym-

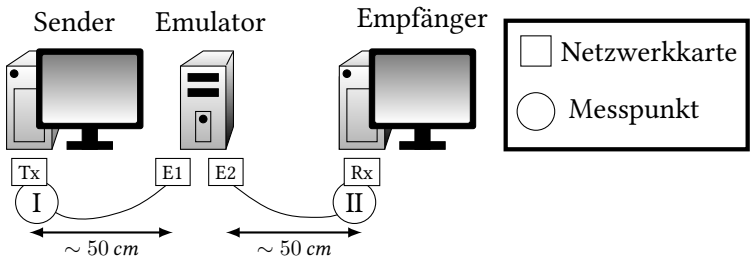


Abbildung 5.2: Angepasster Ansatz des eigenen Versuchsaufbaus mit zwei Messpunkten.

metrischen Aufbaus ist die alternative Halbierung der RTD nur bedingt anwendbar. Bei den geplanten Untersuchungen soll der Emulator an dessen Leistungsgrenzen geführt und nicht durch eine zusätzliche Rückrichtung beeinflusst werden. Auch können einige der zu betrachtenden Emulatoren asymmetrische Topologien nachbilden. Daher sollte der eigene Versuchsaufbau eine Synchronisation der Zeitgeber berücksichtigen. Dies kann beispielsweise durch einen zusätzlichen *Network Time Protocol* (NTP)-Server entweder eigenständig oder als Dienst auf einem der Versuchsrechner bewerkstelligt werden. Um nicht den Probeverkehr zu unterbrechen, könnte eine zusätzliche Direktverbindung zwischen dem Sender und Empfänger über zwei weitere Netzwerkkarten erstellt werden. Dennoch entstehen auch in diesem Aufbau Abweichungen zwischen den beiden Stationen, da jede NTP-Aktualisierung einem Verarbeitungs-Offset und einem Anfrage-Antwort-Umlauf unterliegen. Letzterer kann zwar durch die direkte Verbindung vernachlässigt werden, der Offset jedoch, sorgt für variierende Abweichungen im Mikro- bis Millisekundenbereich. Im RFC [Mil92, S. 100] sind entsprechende Formeln und Erklärungen zu finden.

Die Lösung für dieses Problem ist die Zusammenlegung von Sender und Empfänger auf einem einzelnen Rechner. Dadurch teilen sich beide Rollen den selben Zeitgeber und arbeiten somit synchron. Der bisherige Versuchsaufbau in Abb. 5.2 wird endgültig angepasst und ist schematisch in Abb. 5.3 zu sehen. Der Sender und der Empfänger erhalten dabei jeweils eine eigene Netzwerkkarte, um zum einen die beiden Verbindungskabel des Emulators aufzunehmen und so auf Hardwareebene zwei eigenständige Subnetze erstellen zu können und zum anderen eine gegenseitige Beeinflussung zu verringern. Gleichzeitig verfügt der Messrechner für eine parallele Verarbeitung von gesendeten und empfangenen Paketen einen Mehrkern-Prozessor. Die beiden Messpunkte zur passive Paketaufzeichnung werden auf dem Messrechner positioniert und durch Software realisiert.

5.1.2 Softwaretechnischer Aufbau

Die Grundkomponenten aus softwaretechnischer Sicht sind in Abbildung 5.4 dargestellt und bestehen im Wesentlichen aus einer Versuchssteuerung, einem Paketgenerator, zwei Paketerfassungen sowie einer Analyse- bzw. Berechnungseinheit.

Alle weiteren Objekte in der Abbildung dienen der Versuchskonfiguration, Datengewinnung und -haltung.

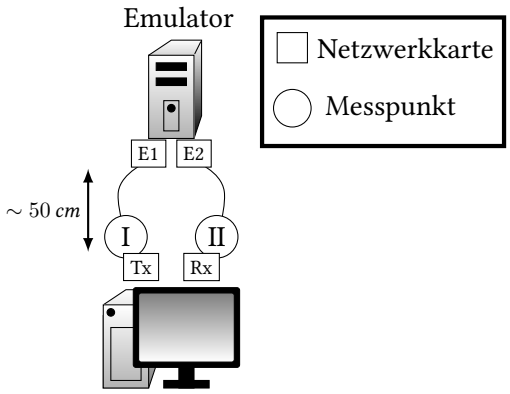


Abbildung 5.3: Hardware-technischer Versuchsaufbau mit zusammengelegtem Sender und Empfänger.

Versuchskonfiguration Die Voruntersuchungen und die eigentlichen Messungen basieren auf mehreren unterschiedlichen Versuchsparametern. Beispielsweise werden die Datenraten bei verschieden Paketgrößen untersucht. Somit bietet es sich an, die Abfolge und die Parameter der verschiedenen Versuche als Konfigurationsdatei zu hinterlegen. An dieser Stelle kann eine GUI zur einfacheren Eingabe der Vorgaben eingesetzt werden. Weiterhin sind für den Paketgenerator und die Paketerfassung Netzwerkeinstellungen wie z. B. die Zuordnung der IP-Adresse und die Angabe des Gateways wichtig. Da auch hier mehrere Angaben benötigt werden, stellt eine Konfigurationsdatei eine komfortable Eingabemöglichkeit dar, welche ebenfalls mittels GUI unterstützt werden kann.

Datengewinnung Aus den versendeten und empfangenen Paketen werden jeweils der Zeitstempel und die zuvor eingefügten Paketnummer mittels passiver Paketerfassung extrahiert. Die Zeitstempel umfassen zum einen den Sekundenanteil und zu anderen den Mikrosekundenanteil des Sende- bzw. Empfangszeitpunktes. Eine geringere Auflösung ist aufgrund der verwendeten Zeitgeber nicht möglich.

Die Zeitstempel werden für die Analysen der Verzögerungen und Datenraten verwendet. Die Paketnummern dagegen bilden die Grundlage für Untersuchungen im Bereich der Paketeffekte. Die Zeitstempel und die Paketnummern sind daher die Rohdaten für alle weiteren Analysen.

Datenhaltung Eine sofortige Ausgabe der Ergebnisse nach jedem Probedurchlauf ist nicht gefordert. Daher lassen sich die Messungen und die Auswertung separat hintereinander ausführen. Dazu ist jedoch eine Datenhaltung nötig, um die erfassten Paketdaten bis zur Auswertung zu speichern. Bei mehreren Versuchen mit mehreren Probedurchgängen sammeln sich Rohdaten in Größenordnungen von mehreren Gigabyte an. Als Beispiel sollen in einem einzelnen Versuch Paketraten im Intervall von $1\,KPkt/s$ bis $200\,KPkt/s$ in $10\,KPkt/s$-Schritten mit jeweils 100 Probedurchläufen untersucht werden. Allein bei der letzten Paketrate ergeben sich für die

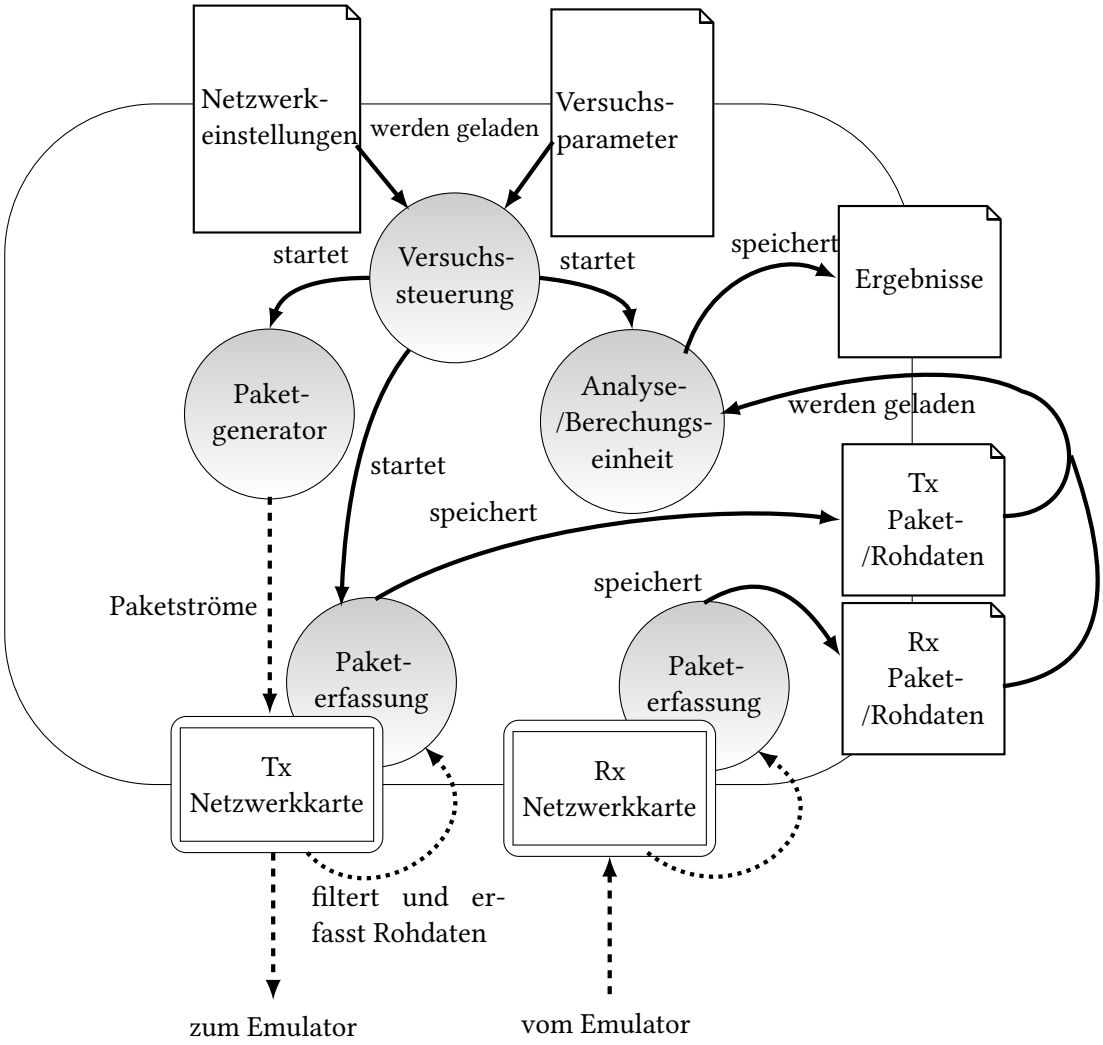


Abbildung 5.4: Softwaretechnischer Versuchsaufbau.

Rohdaten insgesamt (Paketnummern und Zeitstempel werden als Integer-Werte mit jeweils 4 Byte angenommen):

$$\text{Paketanzahl} \cdot (\text{Paketnr.} + \text{Sekundenanteil} + \text{Mikrosekundenanteil}) \cdot \text{Laufzeit}$$

$$= 200\,\frac{\text{KPkt}}{\text{s}} \cdot 12\,\frac{\text{Byte}}{\text{Pkt}} \cdot 100\,s = 229\,\text{MByte}$$

Für den gesamten Versuch ergeben sich insgesamt:

$$100\,s \cdot 12\,\frac{\text{Byte}}{\text{Pkt}} \cdot (1\frac{\text{KPkt}}{s} + \sum_{x=2}^{20}(x \cdot 10\,\frac{\text{KPkt}}{s})) = 2\,\text{GByte}$$

Diese Größe kann von aktuellen Systemen noch im Arbeitsspeicher verarbeitet werden. Ändern sich jedoch die Vorgaben und es wird beispielsweise eine geringere Schrittweite von 1 $KPkt/s$ gefordert, so erhöht sich der Speicherbedarf auf 22 *GByte*. Eine Datenhaltung im Arbeitsspeicher ist daher nur bei Systemen mit mehr als 22 *GByte* RAM möglich. Der zur Messung vorgesehene Rechner besitzt 8 *GByte*, daher werden die Rohdaten für jeden einzelnen Probedurchlauf zunächst als Datei auf der Festplatte gespeichert und zur anschließenden Auswertung je Probedurchlauf geladen. Dieses Vorgehen bietet den Vorteil der Kapselung von Messung sowie Auswertung und gestattet außerdem flexiblere Versuchsszenarien.

Versuchssteuerung Für den automatisierten Ablauf der Experimente wird eine Steuerung eingesetzt, welche Konfigurationen bzgl. Netzwerkeinstellungen und Versuchsparameter aufnimmt und anhand dieser, Aufträge für den Paketgenerator erstellt. Jeder Auftrag entspricht dabei einem Versuch. Ein Versuch besteht aus Probeströmen, welche sich untereinander in bestimmten Stromeigenschaften, wie z. B. Paketgröße, unterscheiden. Ein Probestrom kann mehrmals in einer vorgegebenen Anzahl an Probedurchläufen versendet und analysiert werden. Abbildung 5.5 veranschaulicht die Einteilung.

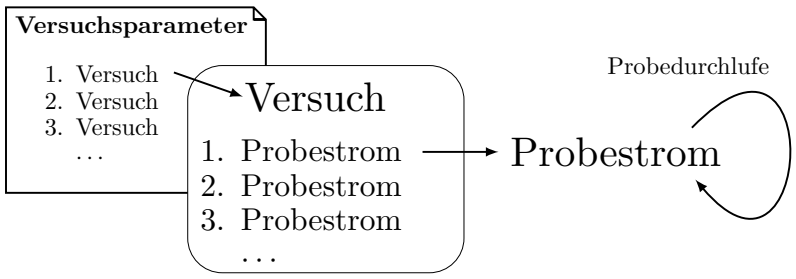


Abbildung 5.5: Einteilung der Versuchsparameter in Probeströme und zugehörige Probedurchläufe.

Zudem koordiniert die Versuchssteuerung den Ablauf der Messungen und der Auswertungen. Dazu gehört die richtige Startreihenfolge von Paketerfassung sowie -versendung und die anschließende Auswertung des Versuchs. In Abbildung 5.6 ist ein entsprechendes Aktivitätsdiagramm zu sehen. Zuerst werden die Prozesse der Paketerfassungen für den Sender und Empfänger gestartet, um so sicher zustellen, dass auch die ersten Pakete aufgezeichnet werden. Anschließend entnimmt die Steuerung aus der Liste der Aufträge einen einzelnen Auftrag und übergibt diesen an den Paketgenerator. Nachdem dieser alle Probeströme mit der vorgegebenen Anzahl an Probedurchläufen versendet hat, beendet die Versuchssteuerung die Paketerfassung und startet die Auswertung des Auftrages. Nach der Auswertung wird mit dem nächsten Auftrag in der Liste fortgefahren und der beschriebene Ablauf wiederholt.

Paketerfassung Die passive Paketerfassung erfolgt parallel für alle aus- sowie eingehenden Pakete und wird in separaten Prozessen gestartet. Jedem Prozess wird eine der beiden Netzwerkkarten zugeordnet. Die ausgehenden Pakete sind für die anschließende Berechnung der

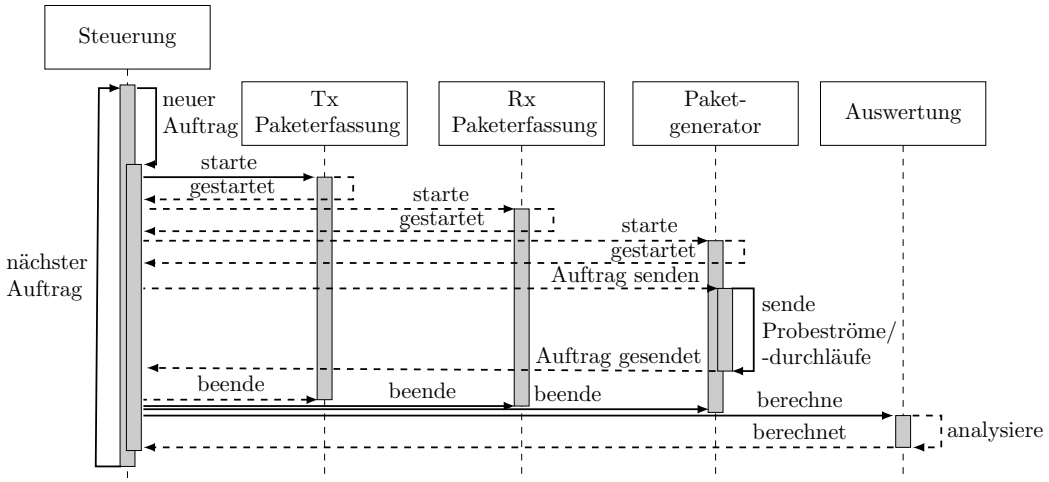


Abbildung 5.6: Sequenzdiagramm der Versuchssteuerung bzgl. der Startreihenfolge der verschiedenen Grundkomponenten.

OWD sowie ΔOWD^j wichtig. Zusätzlich kann so die tatsächliche Senderate ermittelt werden, da durchaus Abweichungen von der geforderten Rate entstehen, welche sonst fälschlicher Weise dem Emulator zugeschrieben werden.

Um die Verarbeitungsgeschwindigkeit bei der Erfassung zu erhöhen, reagieren beide Netzwerkkarten, anders als im sonst für passive Datenerfassung üblichen „promiscuous mode“, nur auf Pakete, welche wirklich nur für diese bestimmt sind. Außerdem werden nur die ersten Bytes mit den benötigten Informationen und der Header eines jeden Paketes erfasst, der Rest wird verworfen. Filter sorgen während der Erfassung dafür, dass versuchsfremde Pakete nicht mit in die Rohdaten gelangen.

Genutzt wird dafür eine eigene Implementation mit Hilfe von angepassten LIBPCAP-Bibliotheken [13l], da für die Auswertung nur bestimmte Paketdaten benötigt werden und sich die eigentliche Datengewinnung leichter in die Paketerfassung integrieren lässt. Programme wie TCPDUMP bietet mehr Funktionen als benötigt und speichern dagegen die Nutzdaten sowie die Header, wodurch eine zusätzliche Extraktion der eigentlichen Daten notwendig wird. Eine weitere Einschränkung wird durch deren Portabilität verursacht. So stehen beispielsweise unter WINDOWS-basierten Systemen Funktionen zur effizienteren Paketerfassung zu Verfügung, welche jedoch nicht von UNIX-artigen Umgebungen unterstützt werden. Daher werden stattdessen die auf beiden Systemen verfügbaren Standardfunktionen verwendet.

Paketgenerator Die Präzision und die Leistungsfähigkeit des Generators legen den möglichen Betrachtungsbereich bei den Messungen fest. So müssen die zeitlichen Abstände zwischen zwei aufeinanderfolgenden Paketen mit einer Auflösung im Mikrosekundenbereich exakt eingehalten werden, um eine konstante Paket- bzw. Datenrate von mehr als $1\,\mathrm{KPkt/s}$ bzw. $1 MBit/s$ (bei L=1500 Byte) zu gewährleisten. Zudem müssen die in Tabelle 4.3 aufgelisteten Anforderungen FI.4 bis FI.6 erfüllt werden. D.h. der Paketgenerator muss Einstellungen bzgl. der Paketgröße und der Sende- sowie Empfangsports unterstützen.

Es existieren bereits viele Hardware- und Software-basierte Generatoren, die im Bereich bis 100 *MBit/s* akkurat arbeiten und diese Einstellungen unterstützen. Eine Übersicht ist auf den Internetseiten [13u] und [13v] zu finden. Bei Datenraten jenseits der 100 *MBit/s* geraten viele Software-Lösungen jedoch an ihre Grenzen. So erreicht beispielsweise der bekannte und viel genutzte D-ITG maximal eine Datenrate von 612 *MBit/s* (Quelle: [Nap08]). Für Paketraten, welche in den Bereichen liegen, bei denen Emulatoren anfangen, Pakete zu verwerfen, müssen Paketabstände von 5 μs bis 2 μs erreicht werden, was den Raten von 200 KPkt/s bis 500 KPkt/s entspricht. Kommerzielle Paketgeneratoren bieten diese hohe Leistungsfähigkeit und basieren zumeist auf Hardware-Lösungen. Beispiele dafür sind der „LANforge-FIRE Traffic Generator“ [13b] oder „Ixia Anue 3500“ [13i]. Allerdings stehen solche Produkte nicht zur Verfügung.

Programme, welche die bisherigen Anforderungen umsetzen und Quelloffen verteilt werden, sind z. B. Ostinato[13t] und Iperf [13h]. Dennoch fehlen auch bei diesen wichtige Einstellungsmöglichkeiten, um alle geforderten Metriken aus Tabelle 4.3 ermitteln zu können. So kann bei Iperf die Paketrate und der zeitliche Sendeabstand nur indirekt über die Bitrate angegeben werden. Auch benötigt jedes Paket eine Sequenznummer sowie die aktuelle Paketstrom- und Probedurchlaufnummer, um beispielsweise Analysen bezüglich der Neuordnung und Dopplung zu ermöglichen. Diese Konfigurationsmöglichkeit findet sich jedoch in keinem der erwähnten Generatoren. Folglich muss entweder eine Anpassung am jeweiligen Programm vorgenommen oder ein eigener Paketgenerator entwickelt werden. Erstere Möglichkeit erfordert eine intensive Einarbeitung in den Quellcode. Für die andere Option kann dagegen ein bereits vorhandener Paketgenerator modifiziert werden.

In einer vorangegangenen Arbeit [Büs13] wurde für das Messtool Nora ein Generator entwickelt, welcher alle notwendigen Einstellungsmöglichkeiten umsetzt. Jedoch liegt der minimal mögliche Paketabstand unter beispielsweise „Windows 7“ bei 16 μs und damit bei einer maximalen Paketrate von 62,5 KPkt/s. Der Grund dafür liegt in der Art der Generierung der Pakete. Jedes Paket eines Probedurchlaufs wird vor dem Versenden zunächst in der Anwendungsebene mit den nötigen Informationen versehen und anschließend über einen *Socket* an den Kernel übergeben. Abbildung 5.7 veranschaulicht den Ablauf. Dadurch entsteht zum einen eine Abhängigkeit von der Interrupt-Rate des Nutzerbereichs und zum anderen entstehen Verzögerungen durch die periodische Speicherallokation sowie durch die Übergabe an den Kernel.

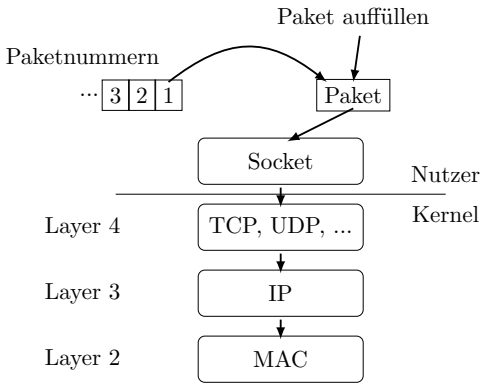


Abbildung 5.7: Die Paketgenerierung in der vorangegangenen Arbeit [Büs13].

Die Lösung ist, alle Pakete eines Probedurchlaufes vor dem Versenden zu generieren und im Kernelbereich zu puffern (vgl. Abbildung 5.8). Anstelle der Sockets in der Anwendungsebene, werden beispielsweise unter Unix-artigen Systemen direkt die Paket-Sockets in der Ebene des Netzwerkkartentreibers (Layer 2) verwendet. Auch in Windows-basierten Systemen ist dieser Ansatz möglich und es existieren bereits entsprechende Funktionen dafür, wodurch der Zeitabstand auf 1 μs reduziert werden kann. Dieses Vorgehen wird z. B. von der bekannten WinPcap-Bibliothek verwendet.

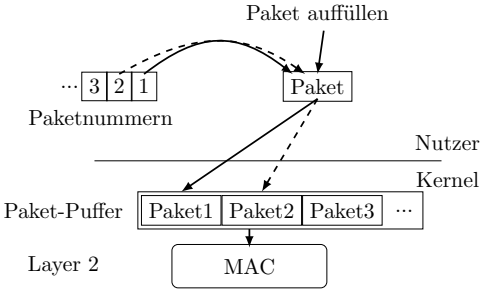


Abbildung 5.8: Die verbesserte Paketgenerierung in dieser Arbeit.

Da kein bestimmtes Betriebssystem gefordert wird, kann auf diese Windows-Funktionen zurückgegriffen werden. Damit entsteht zwar eine Betriebssystemabhängigkeit, jedoch fällt der zeitliche Umfang für die Modifizierung des eigenen vorhandenen Paketgenerators geringer aus als die Einarbeitung in fremden Quellcode. Für spätere Versionen des Paketgenerators kann auch diese Abhängigkeit, durch eine Implementierung der Unix-eigenen Funktionen beseitigt werden.

Analyse-/Berechnungseinheit Die Auswertung geschieht im Anschluss an die Messungen eines einzelnen Versuches und baut auf mehreren Algorithmen auf, um zunächst aus den Rohdaten Einzelergebnisse und anschließend die geforderten statistischen Werte FA.1 bis FA.7 zu ermitteln. Für jeden Probedurchlauf wird im Laufe der Messungen eine Datei angelegt, welche die Paketnummern und die zugehörigen Zeitstempel enthält. Diese Rohdaten bilden die Grundlage für alle nachfolgenden Analysen. Berechnet werden die Werte der Metriken FM5.1 bis FM5.7 (vgl. Tab. 4.3) unter Zuhilfenahme von Algorithmen aus der vorangegangenen Arbeit [Büs13]. In dieser wurde ein Messwerkzeug für Ende-zu-Ende Verbindungen entwickelt, welches bis auf die OWD alle geforderten Metriken berechnen kann. Daher muss neben einigen wenigen Anpassungen, lediglich ein Algorithmus für die OWD zugefügt werden. Dieser wird im nächsten Abschnitt zur Vorgehensweise beschrieben.

Die Ergebnisse dieser Berechnungen werden zum einen für den einzelnen Probedurchlauf und zum anderen für den Probestrom als Datei gespeichert. Nach dem letzten Probestrom eines Versuchs erfolgt eine gesonderte Auswertung aller gesammelten Ergebnisse. Der genaue Ablauf der Auswertung und die Berechnungen sind Teil des nächsten Abschnittes.

5.2 Vorgehensweise

Die Vorgehensweise gliedert sich in drei Phasen. In der ersten Phase werden alle Vorfelduntersuchungen durchgeführt, um zum einen den Einfluss des Versuchsaufbaus auf die Messungen

und zum anderen den Paketpuffer des Emulators zu bestimmen. Die zweite Phase beinhaltet die eigentlichen Messungen, welche die Werte für die dritte Phase, die Auswertung, liefert.

5.2.1 Vorfelduntersuchungen

Im Vorfeld der eigenen Untersuchungen werden die Puffergröße, die Grundverzögerung und die MLFFR des Versuchsaufbaus bestimmt. Auf eine automatisierte Betrachtung der Taktrate durch z. B. eine modifiziertes PING (vgl. Abschnitt 3.3.1) wird verzichtet, da die Informationen auch manuell abgerufen werden können. Die gewonnenen Werte sind wichtig, da der Versuchsaufbau selbst Einflüsse und Beschränkungen auf die Ergebnisse der eigentlichen Messungen verursacht. Die Beschränkungen bzw. Obergrenzen müssen bekannt sein, da diese sonst fälschlicherweise dem Emulator zu geschrieben werden könnten.

Puffergröße In [Min+11] wurde die Auswirkung der Puffergröße auf die Stabilität der Paketraten gezeigt. KAUNET als Beispiel hatte im Vergleich zu NETEM einen halb so großen Puffer und zeigte dementsprechend eine größere Streuung in den Ergebnissen. Für die eigenen Untersuchungen ist die Puffergröße daher ebenfalls interessant. Dazu wird eine Datenratenbeschränkungen am Emulator eingestellt und ein Paketstrom mit konstantem Paketabstand verwendet. Der Paketstrom muss den Emulator dabei komplett auslasten oder sogar überlasten. D. h. das Verhältnis aus eingestellter Rate und tatsächlich emulierter Rate muss größer gleich Eins betragen. Gemessen werden neben der durchschnittlichen emulierten Datenrate auch die durchschnittliche Verzögerung, um so die Puffergröße nach Gleichung 3.3 zu erhalten. Das folgende Beispiel soll das Vorgehen verdeutlichen:

> Am Emulator wird eine Datenratenbeschränkung von z. B. $R_{gefordert} = 1\,\mathrm{MBit/s}$ eingestellt. Um den Emulator auszulasten bzw. zu überlasten, müssen die generierten Paketströme ebenfalls mit einer Rate von mindestens $R_{eingehend} = 1\,\mathrm{MBit/s}$ versendet werden. Als Indikator für die Auslastung wird das Verhältnis zwischen der tatsächlichen emulierten Rate $R_{emuliert}$ und $R_{gefordert}$ beobachtet. Für dieses muss gelten:
>
> $$\frac{R_{gefordert}}{R_{emuliert}} \geq 1$$
>
> Um eine solche Datenrate beispielsweise für die kleinst mögliche Paketgröße von $64\,\mathrm{Byte}$ zu erreichen, wird ein Paketabstand von:
>
> $$\Delta t = \frac{64\,\mathrm{Byte}}{1\,\mathrm{MBit/s}} = 512\mu s$$
>
> benötigt. Dieser kann auf $500\mu s$ abgerundet werden, um sicherzustellen, dass der Emulator stets ausgelastet ist. Gemessen wird die emulierte Datenrate $R_{emuliert}$ und die OWD am Empfänger für 10000 Pakete pro Paketgröße. Für jede Paketgröße wird nach den Messungen die durchschnittliche emulierte Rate
>
> $$\bar{R}_{emuliert} = \frac{\sum_{i=0}^{9999} R^i_{emuliert}}{10000}$$

sowie die durchschnittliche Verzögerung

$$\bar{D} = \frac{\sum_{i=0}^{9999} \mathrm{OWD}^i}{10000}$$

berechnet. Diese Werte werden abschließend in die Gleichung 3.3 eingesetzt, um so die Puffergröße P zu erhalten, wobei L der aktuellen Paketgröße von $64\,\mathrm{Byte}$ entspricht:

$$P_L = \frac{\bar{D} \cdot \bar{R}_{emuliert}}{L}$$

Grundverzögerung Jeder von den vorgestellten Emulatoren arbeitet mit einem anderen Betriebssystem zusammen. Teilweise werden diese, anstelle der üblichen Installation auf der Festplatte, per Live-System gestartet. Dadurch ergeben sich unterschiedliche Voraussetzungen, welche sich u. a. besonders durch verschiedene Grundverzögerungen äußern.

Diese Verzögerungen werden allein durch den Versuchsaufbau, daher ohne den gestarteten Emulator verursacht. Um eine weitestgehend vom Betriebssystem unabhängige emulierte Verzögerung zu erhalten, wird dieses Grundrauschen bei den eigentlichen Messungen berücksichtigt. Für die Ermittlung der Grundverzögerung wird lediglich das jeweilige Betriebssystem auf dem Rechner des Emulators gestartet. Die Emulation selbst bleibt für diese Voruntersuchung deaktiviert.

Gemessen werden mehrere Paketströme mit denjenigen Eigenschaften, welche auch in den eigentlichen Messungen verwendet werden. D. h., es werden verschiedene Paketgrößen und unterschiedliche Paketabstände überprüft. Sollen beispielsweise zehn ausgewählte Paketraten für fünf verschiedene Paketgrößen gemessen werden, so werden im Vorfeld die Grundverzögerung für ebenfalls diese Raten und Paketgrößen ermittelt. Die Grundverzögerung ist dabei die durchschnittliche Verzögerung aller OWDs des jeweiligen Paketstromes. Die Ergebnisse dieser Voruntersuchungen werden von den ermittelten Verzögerungen in der eigentlichen Messung abgezogen, um so die tatsächliche Verzögerungen, welche durch den Emulator verursacht werden, zu erhalten.

Maximale verlustfreie Paketrate Neben der Grundverzögerung, muss ebenfalls die maximal mögliche Datenrate innerhalb der Versuchsumgebung ermittelt werden. Auch in diesem Fall geschieht die Untersuchung ohne den Emulator. Gemessen wird die MLFFR des Versuchsaufbaus, um eine Obergrenze für die eigentliche Messung zu erhalten. Dadurch wird sichergestellt, dass es zu keiner Verwechslung zwischen der MLFFR des Emulators und der maximalen verlustfreien Paketrate des Versuchsaufbaus kommt. Im optimalen Fall beschränkt der Versuchsaufbau nicht die Leistungsfähigkeit des zu testenden Emulators und die Obergrenze wird nicht erreicht. Sollte dennoch der Fall eintreten, dass diese Obergrenze in den eigentlichen Messungen erreicht wird, so wird dieses in den Ergebnissen bekannt gemacht.

5.2.2 Messungen

Nach den Voruntersuchungen beginnt die zweite Phase; die eigentliche Messung. Diese liefert die gesuchten Werte bezüglich der Genauigkeit und Leistungsfähigkeit.

Allgemeine Durchführung Die allgemeine Vorgehensweise ist dabei, den Emulator mit bestimmten Vorgaben aus den Bereichen:

1. *Paketeffekte*
2. *Verzögerungen*
3. *Datenraten*

zu konfigurieren und Probeströme mit unterschiedlichen Eigenschaften bezüglich:

(a) der Paketgröße
(b) des Paketabstandes
(c) der Paketanzahl

vom Sender über den Emulator an den Empfänger zu versenden. Gemessen werden die Paketnummer und der Zeitstempel jedes gesendeten sowie empfangenen Paketes. Aus den Messwerten sollen in der anschließenden Auswertung Wertebereiche, Auflösungen, Abweichungen und Wertstabilitäten ermittelt werden. Besitzt der Emulator außerdem die Möglichkeit mehrere Emulationseinheiten bzw. Pfadabschnitte nachzubilden, so werden die Vorgaben 1. bis 3. mittels verschiedener Stromeigenschaften (a) bis (c) bei ansteigender:

(i) serieller Pfadverkettung
(ii) paralleler Pfadverkettung

gemessen, um zusätzlich Aussagen über die Skalierbarkeit zu erhalten.

Die Parameter einer Messung ergeben sich aus der Kaskadierung einer zu emulierenden Vorgabe aus den Bereichen 1. bis 3. mit jeweils bestimmten Stromeigenschaften (a) bis (c) sowie für jeweils aufsteigende (i) bzw. (ii). Zur Veranschaulichung lässt sich der in Abbildung 5.9 dargestellte Baum aufspannen. Gleichzeitig lässt sich anhand des Baumes der Begriff der Genauigkeit und Leistung skizzieren. Die Messungen, welche auf den vertikal verknüpften Parametern beruhen, liefern die Ergebnisse für die Genauigkeitsbetrachtung einer Emulation. Während die gemessene Umsetzung der Einstellungen in horizontaler Richtung, die dabei erreichten Leistungen anzeigen. Beispielsweise wird bei der Messung einer bestimmten Verzögerung in mehreren Probedurchläufen die Datenrate erhöht. Der Messwert der jeweiligen tatsächlich emulierten Verzögerung gibt die Genauigkeit und die dabei ebenfalls gemessene Datenrate gibt die Leistung des Emulators an. Die genauen Messparameter werden im Kapitel zur Umsetzung der Messung erläutert.

Konkrete Durchführung Die konkreten Messungen unterteilen sich in die Ermittlung der erreichbaren Granularität der einstellbaren Paketeffekte, Verzögerungen sowie Datenraten und in die Untersuchung des jeweiligen Wertebereiches unter der Berücksichtigung von Abweichungen, Wertstabilitäten sowie Skalierungen. Für beide Unterteilungen werden die gleichen Paketstromeigenschaften verwendet, da die Ergebnisse der Messungen zur Granularität die Grundlage für die Untersuchungen zum Wertebereich bilden. Daher wird zunächst die kleinstmögliche Verzögerung, Datenrate sowie Effektrate bei ausgewählten Paketgrößen, Paketabständen und Pfadlängen ermittelt.

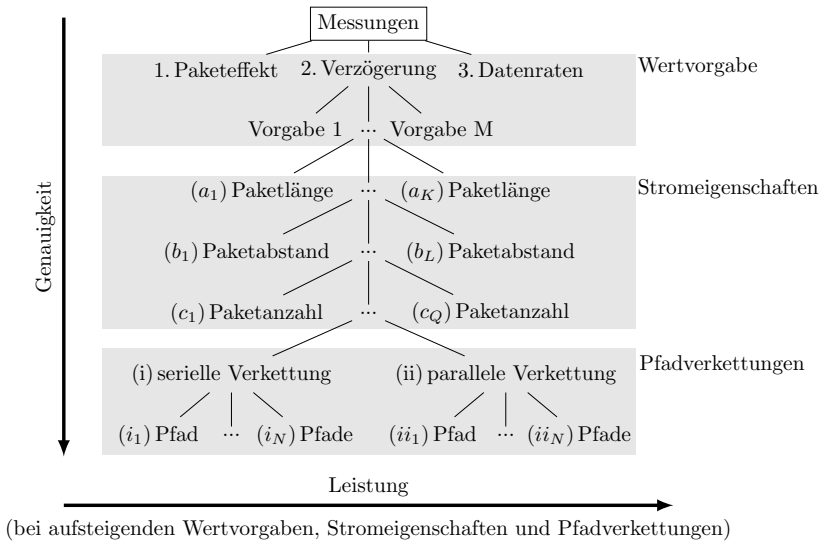


Abbildung 5.9: Allgemeines Schema der Parameterverknüpfungen für die Messungen.

Granularität: Die kleinstmögliche Effektrate wird durch eine schrittweise Verringerung der Wertvorgabe mit entsprechender Erhöhung der Paketanzahl je Probestrom ermittelt (vgl. Tabelle 5.1). Denn es gilt beispielsweise bei einer Vorgabe von $1\,\%$ Paketverlust, -verdopplung oder -neuordnung, muss ein Paket von 100 gesendeten Paketen in einem Paketstrom von der Manipulation des Emulators betroffen sein. Bei $0,1\,\%$ muss ein Paket von 1000 Paketen, bei $0,01\,\%$ eines von 10000 Paketen usw. manipuliert werden. Bei der Messung muss jedoch die Eigenheit des Pseudozufallsgenerators bei einigen Emulatoren, nur für lange Paketfolgen die richtige Prozentzahl an Paketen zu beeinflussen, berücksichtigt werden. Daher werden alle Messungen ohne vorzeitigen Abbruch durchgeführt und erst hinterher die Abweichung kontrolliert. Um Abweichungen und Ausreißern entgegen zu wirken, werden für jeden Probestrom mehrere Probedurchläufe durchgeführt und die Ergebnisse gemittelt. Die kleinstmögliche Rate ist erreicht, wenn entweder der Probestrom eine Länge annimmt, dessen Generierung nicht mehr möglich ist oder die Abweichungen außerhalb einer bestimmten Toleranz liegen.

Die kleinstmögliche Verzögerung ist abhängig von der verwendeten Taktrate des Emulators. Ist der Wert der Taktung bekannt, so dient dieser als Ausgangspunkt für die Untersuchung. Alternativ wird eine geringe Verzögerung als Emulationsvorgabe gewählt. Die Vorgabe wird am Emulator stufenweise verringert und jeweils mit einer festen Anzahl an Paketen untersucht(vgl. Tabelle 5.1). Mehrere Probedurchläufe sind nicht nötig, da die Verzögerung jedes Paket in einem Paketstrom beeinflusst und bei entsprechender Stromlänge die Gewichtung von Ausreißern verringert wird. Die kleinstmögliche Verzögerung ist erreicht, wenn entweder der Vorgabewert den Wert der gemittelten Grundverzögerung des Versuchsaufbaus annimmt oder die Abweichungen außerhalb einer bestimmten Toleranz liegen.

Die kleinstmögliche emulierbare Datenrate wird durch die kontinuierliche Verringerung der Wertvorgabe am Emulator bei gleichbleibender Paketrate und -anzahl je Paketstrom ermittelt. Es wird daher untersucht, wie stark die emulierten Datenraten von den Vorgaben bei einer bestimmten generierten Datenrate abweichen. Auch hier ist lediglich ein Probedurchlauf pro

	Paketeffekte	Verzögerungen	Datenraten
Paketanzahl	steigend	konstant	konstant
Probedauer	–	–	erhöht
Probedurchläufe	mehrere	einzeln	einzeln

Tabelle 5.1: Allgemeine Probestromeigenschaften je Emulationsvorgabe für die einzelnen Bereiche bei der Untersuchung der Granularität und des Wertebereiches. Pro Vorgabe werden dabei mehrere Paketabstände und -größen untersucht.

Probestrom nötig(vgl. Tabelle 5.1). Um jedoch den Einfluss von stark abweichenden Werten zu verringern, wird die Zeitdauer eines Probedurchlaufs wesentlich erhöht. Die kleinste emulierbare Datenrate ist gefunden, wenn alle nachfolgenden gemittelten Raten eine bestimmte Abweichung überschreiten.

Wertebereich: Für die Ermittlung des Wertebereichs werden die Vorgaben für die Paketeffekte, Verzögerungen und Datenraten am Emulator Stufenweise erhöht. Die Schrittweite bzw. die Anzahl der Nachkommastellen pro Schritt richtet sich nach den Ergebnissen zur Granularität. Die untere Grenze des Wertbereichs ist die kleinstmögliche emulierbare Wertvorgabe. Zur Gewinnung der oberen Grenze werden zunächst alle Stufen durchlaufen und die zugehörigen Messergebnisse gespeichert. Die endgültige Auswertung der Messwerte findet erst nach Abschluss der letzten Stufe statt. Somit werden auch Vorgaben bzgl. ihrer Abweichung und Stabilität überprüft, welche bereits außerhalb des möglichen Wertebereichs liegen. In der Phase der Auswertung wird anschließend mit bestimmten Toleranzen die obere Grenze ermittelt. Das Ziel bei diesem Vorgehen ist es, Vergleichsmöglichkeiten zwischen allen Emulatoren zu schaffen, um beispielsweise Aussagen über Schwankungen zu einer bestimmten emulierten Vorgabe bei gleichen Voraussetzungen treffen zu können. Ein vorzeitiger Abbruch wegen Toleranzüberschreitung würde diese Möglichkeit verhindern. Die Vorgehensweise bei den Paketeffekten, Verzögerungen und Datenraten in Bezug auf die Anzahl der Pakete und Probedurchläufe sowie Zeitdauer gleicht dem Ablauf bei der Granularität (vgl. Tabelle 5.1).

5.2.3 Auswertung

Die Rohdaten der einzelnen Probedurchläufe eines Versuches werden während der Messungen jeweils in einer Datei gespeichert und müssen für die abschließende Auswertung in der dritten Phase analysiert werden. Die Analyse liefert die Einzelwerte zu den geforderten Metriken FM.7.1 bis FM.7.7 für jeden einzelnen Probedurchlauf und die statistischen Werte FA.1 bis FA.7 für ebenfalls alle einzelnen Probedurchläufe eines Probestromes sowie für den Probestrom selbst. Die für die Ermittlung der Einzelwerte benötigten Algorithmen werden bis zu einem gewissen Grad aus der vorgehenden Arbeit [Büs13] zum Messtools Nora übernommen. Dazu zählen die Algorithmen für die Ermittlung der Datenraten, ΔOWD^j und Paketeffekte. Wobei letztere Algorithmen um die Möglichkeit erweitert werden müssen, konkrete Paketnummern auszugeben, um zum einen die Genauigkeit bei musterbasierten Emulatoren wie KauNet prüfen zu können und zum anderen für die Visualisierung von z. B. Intervallen in denen ein Emulator Pakete gewollt bzw. ungewollt verwirft, verdoppelt oder neu ordnet.

Die genaue Vorgehensweise für die Analyse der einzelnen Metrikwerte und die Berechnung der statistischen Werte werden dazu im Folgenden beschrieben.

Paketeffekte Die Algorithmen für die Ermittlung der Paketverluste, -dopplungen und -neuordnungen sind in NORA dafür ausgelegt worden, bereits während des Versuches Ergebnisse zu liefern. Anstelle der Ausgabe der konkreten Paketnummern wird dabei lediglich die Anzahl der Pakete benötigt, welche entweder verloren, verdoppelt oder neu geordnet wurden. Um jedoch auch die zugehörigen Paketnummern zu erhalten, werden die Algorithmen angepasst.

Paketverluste und -dopplungen werden bisher mit Hilfe von Abständen zwischen zwei aufeinanderfolgenden Paketnummern gezählt. Bei Dopplungen beträgt der Abstand gleich Null. Paketverluste dagegen werden durch Abstände größer Eins gekennzeichnet. Um an die betreffenden Paketnummern zu gelangen, bieten sich drei Möglichkeiten an.

Die erste Möglichkeit besteht darin, bei Abständen ungleich Eins zusätzliche Routinen einzufügen, welche für die Dopplungen einfach die beteiligten Paketnummern speichert bzw. für die Paketverluste die fehlenden Nummern zwischen den zwei aktuellen Paketnummern ermittelt und anschließend speichert. Diese Lösung hat den Nachteil, dass für jede Paketnummer mindestens zwei Berechnungen benötigt werden, unabhängig davon ob ein Paketeffekt aufgetreten ist. Zuerst werden zwei aufeinanderfolgende Nummern voneinander abgezogen und dann wird das Ergebnis bzw. der Abstand mit Eins verglichen.

Bei der zweiten Möglichkeit wird eine Liste L_{erw} mit den erwarteten Paketnummern durchlaufen und das Vorkommen der aktuellen Paketnummer in der gemessenen Paketliste L_{gem} gezählt. Die Paketnummern und deren Anzahl an Vorkommen werden anschließend zusammen gespeichert. Auch hier entstehen Nachteile für die Performanz, da für jede Nummer aus L_{erw}, die gemessene Paketliste durchlaufen wird.

Die dritte Möglichkeit verknüpft die beiden erst genannten Ansätze und reduziert die Mindestanzahl der Berechnungen pro Paketnummer auf einen einzelnen Vergleich. Für das eigene Messkonzept eignet sich diese Möglichkeit mit Blick auf die Performanz daher am besten. Die folgenden Beschreibungen sollen den Algorithmus zusammen mit einem Beispielablauf verdeutlichen.

Wenn beide Listen L_{erw} und L_{gem} gleichermaßen, z. B. aufsteigend, sortiert sind, stimmt die erwartete Paketnummer mit der gemessenen solange überein bis entweder eine Dopplung oder ein Verlust aufgetreten ist. Bei einer Dopplung ist die erwartete Nummer größer als die Gemessene; bei einem Verlust dagegen ist diese kleiner. Stellt man sich beide Liste als Schieberegister vor, so wird L_{gem} bei einer Dopplung solange um eine Paketnummer verschoben bis der erwartete Wert mit dem Gemessenen übereinstimmt, während L_{erw} festgehalten wird. Gleichzeitig wird gezählt wie oft L_{gem} verschoben wird. Bei einem Paketverlust wird L_{gem} festgehalten und stattdessen wird L_{erw} verschoben. Dieses Mal werden die erwarteten Paketnummer gespeichert, um welche L_{erw} für die Übereinstimmung verschoben werden muss. In Abbildung 5.10 ist ein Beispiel dazu gegeben, bei dem zunächst ein einfacher Paketverlust, dann zwei Dopplungen und anschließend ein weiterer Verlust erkannt werden. Im Beispiel werden die Paketnummern $L_{erw} = \{1, 2, 3, 4, 6\}$ erwartet. Zu Beachten ist, dass L_{erw} das fünfte Paket nicht enthält. Das kann passieren, wenn beispielsweise L_{erw} der gesendeten Paketliste entspricht und das fünfte Paket beim Sender verworfen wurde. Gemessen werden die Paketnummern $L_{gem} = \{2, 2, 2, 3, 4\}$. Daher fehlen die Pakete $1, 6$ und das zweite Paket wurde zweimal verdoppelt. Entsprechend wird im ersten Schritt

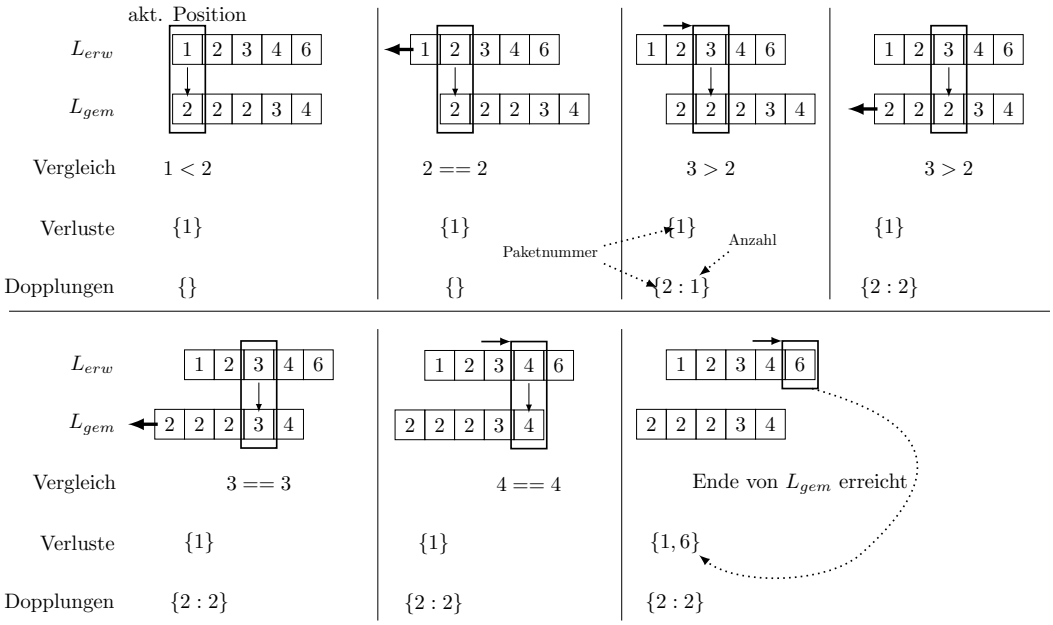


Abbildung 5.10: Beispiel für die Analyse von Paketverlusten und -dopplungen.

das fehlende Paket mit der Nummer 1 entdeckt, da die aktuelle erwartete Paketnummer 1 kleiner als die aktuelle gemessene Paketnummer 2 ist. Der Wert 1 wird gespeichert und L_{erw} wird im nächsten Schritt nach links verschoben, während die aktuelle Position in L_{gem} erhalten bleibt. In dieser Position stimmen beide Paketnummern wieder überein, so dass keine weiteren Behandlung notwendig wird. Im dritten Schritt wird daher die aktuelle Position nach rechts verschoben. Die erwartete Nummer ist in diesem Fall größer als die Gemessene. Somit handelt es sich um eine Dopplung. Die Paketnummer wird zusammen mit einem Zähler gespeichert und L_{gem} wird für den nächsten Schritt nach links verschoben. Hierbei wird ebenfalls eine Dopplung erkannt und der Zähler wird erhöht. Im fünften sowie sechsten Schritt stimmen beide Paketnummern überein und lediglich die aktuelle Position wird verschoben. Gleichzeitig wird das Ende von L_{erw} erreicht, was bedeutet, dass die verbleibende Paketnummer 6 in L_{erw} für einen weiteren Paketverlust steht und dementsprechend zur Liste der Verluste hinzugefügt werden.

Das Ergebnis sind zwei Listen, wobei eine alle Paketnummern der Paketverluste enthält und die andere die Paketnummern zusammen mit der Anzahl der doppelten Pakete beinhaltet.

Paketneuordnungen werden ebenfalls mit einer angepassten Version des diesbezüglichen Algorithmus aus der vorangegangen Arbeit ermittelt. Der Pseudocode mit der Anpassung ist in Abbildung 5.11 zu sehen. Dem ursprünglichen Algorithmus wird im Alternativzweig der Abfrage eine Routine zur Speicherung der Paketnummer (siehe Zeile 9) zugefügt. Andere Anpassungen sind nicht nötig. Um an die neu geordneten Paketnummern zu gelangen, wird in der gemessenen und unsortierten Paketnummernliste jedes Paket mit einem wandernden Maximalwert verglichen. Ist die aktuelle Paketnummer größer als der bestehende Maximalwert, so wird diese Nummer der neue Maximalwert. Ist jedoch der Maximalwert größer als die aktuelle Paketnummer, so ist eine Neuordnung aufgetreten. Die betreffende Nummer wird gespeichert und die nächste

```
1  lokal_max     = 0 #Speichert lokales Maximum
2  zähler        = 0 #Zählt die Pakete au ß erhalb der Reihenfolge
3  nummern_liste = [] #Speichert die Paketnummern der Neuordnungen
4
5  for i = 0 bis länge(paketnummernliste):
6     if lokal_max <= paketnummernliste[i]:
7        lokal_max = paketnummernliste[i]
8     else:
9        nummern_liste.einfügen(paketnummernliste[i])    #<--- Anpassung
10       zähler += 1
```

Abbildung 5.11: Pseudocode: Bestimmung der Anzahl und Paketnummern von neu geordneten Paketen.

Paketnummer wird geprüft. Als Ergebnis erhält man die Anzahl und die Paketnummern der neu geordneten Pakete.

Verzögerungen Die nächsten notwendigen Metriken sind die OWD und ΔOWD^j. Für Letztere kann der Algorithmus komplett, inklusive der Behandlung von Interrupt-Verschmelzungen (engl. Interrupt Coalescence), vom Messwerkzeug Nora übernommen werden. Eine direkte Implementation für die Ermittlung der OWD existiert jedoch nicht in Nora, da das Werkzeug für Ende-zu-Ende Verbindungen ohne Zeitsynchronisation ausgelegt wurde, bei denen Sender und Empfänger auf unterschiedlichen Rechnern installiert werden. Allerdings kann der Algorithmus der ΔOWD^j soweit verändert werden, dass auch die Einwegeverzögerung berechnet werden kann. Anstelle des zeitlichen Abstandes zwischen zwei aufeinanderfolgenden Paketen, wird der Abstand zwischen dem Sende- und Empfangszeitpunkt berechnet. Alle anderen Mechanismen, wie z. B. die Berücksichtigung von Paketverlusten und -dopplungen, werden aus der ursprünglichen Lösung übernommen. Eine schematische Darstellung des angepassten Algorithmus für die OWD-Berechnung ist in Abbildung 5.12 mit Beispielwerten zu sehen.

Nach der Messung eines Probedurchlaufes existiert jeweils eine Liste mit den Rohdaten für den Sender und Empfänger. In einem ersten Schritt wird die Empfängerliste aufsteigend sortiert, wodurch alle Neuordnungen beseitigt werden. Im Beispiel betrifft dieses das fünfte Paket. Anhand der Paketnummern werden anschließend alle Verluste sowie Dopplungen in der Datei des Empfängers mit Hilfe der zuvor beschriebenen Vorgehensweise ausfindig gemacht und vermerkt. Im Beispiel tritt ein Verlust des dritten und eine Dopplung des ersten Paketes auf. Die OWD kann bei verloren gegangenen Paketen nicht berechnet werden, da kein Empfangszeitpunkt vorhanden ist. Daher werden die betreffenden Pakete auch in der Senderliste verworfen. Bei doppelten Paketen gilt, dass immer das erste Vorkommen für die Berechnungen verwendet wird. Alle anderen werden verworfen, da bei diesen kein Sendezeitpunkt existiert. Nach dieser Behandlung werden für die einzelnen OWDs die verbleibenden Zeitstempel des Senders von denen des Empfängers abgezogen. Das Resultat ist eine Liste von OWDs der einzelnen Pakete.

Datenraten Für die Auswertung wird hauptsächlich die versendete und empfangene Paketrate (PPS) bzw. Datenrate (BPS) verwendet. Lediglich für die Betrachtung von emulierten CT wird

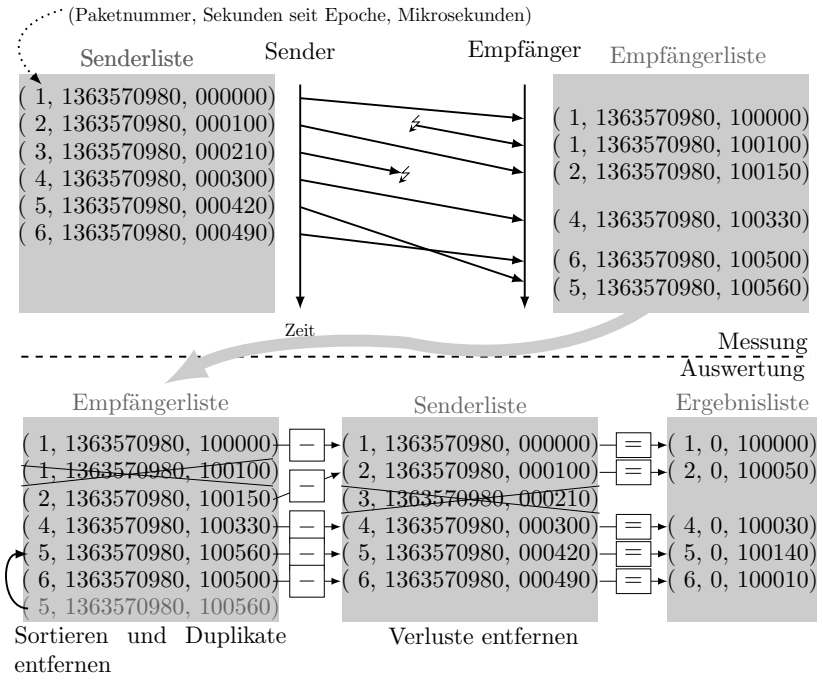


Abbildung 5.12: Beispiel für das Vorgehen und die Berechnung der einzelnen OWD-Werte.

zusätzlich die verfügbare Datenrate A benötigt. Die Vorgehensweise dazu wird komplett von Nora übernommen und beruht auf einer Trendanalyse der aktuellen Datenrate mit Hilfe des aktuellen Paketabstandes. Ist der Paketabstand beim Empfänger gleich dem gesendeten Abstand, so ist die aktuelle Rate kleiner als die verfügbare Datenrate. Die verfügbare Datenrate entspricht derjenigen Rate bei welcher der Paketabstand ansteigt.

Die PPS berechnet sich für die Menge der erfassten Pakete M_p mit $|M_p| > 1$ wie folgt:

$$\mathrm{PPS} = \frac{|M_p| - 1}{t_{end} - t_{start}} \tag{5.1}$$

Dabei entspricht t_{start} dem Zeitstempel des ersten und t_{end} dem des letzten Paketes. Doppelte Pakete gehen mit in die Berechnung ein, während Paketverluste nicht berücksichtigt werden. Bei der Bildung der Summe wird das erste Paket außen vor gelassen, da der Zeitstempel erst nach Erhalt des letzten Bits vergeben wird.

Entsprechend berechnet sich die BPS mit Hilfe der folgenden Gleichung:

$$\mathrm{BPS} = \frac{\sum_{j=2}^{p} L_j}{t_{end} - t_{start}} \tag{5.2}$$

Wobei L der gemessenen Paketgröße des Paketes j in Byte entspricht.

statistische Werte Die Berechnungen der Statistiken erfolgt nach jedem Probedurchlauf auf Grundlage der Metrikwerte, welche mit Hilfe der zuvor genannten Verfahren ermittelt wurden.

Die einzelnen Ergebnisse eines Durchlaufs bilden wiederum die Berechnungsgrundlage für die statistischen Werte des Probestromes.

Die ***Minima und Maxima*** aus einer Menge von Metrikwerten beispielsweise, werden zunächst für alle Probedurchläufe ermittelt und in einer Liste gespeichert. Anschließend werden aus dieser Liste der Gesamt-Maximalwert der Maxima bzw. Gesamt-Minimalwert der Minima des Probestromes gewonnen:

$$\text{Probestrom}^{Metrik} = f^{Metrik}\{f^{Metrik}_{\text{Durchlauf 1}}\{\text{x}_1, \ldots, \text{x}_\text{i}\}, \ldots, f^{Metrik}_{\text{Durchlauf N}}\{\text{x}_1, \ldots, \text{x}_\text{k}\}\}$$

$$f \equiv max() \text{ bzw. } min()$$

Wobei N der Anzahl der Probedurchläufe und i bzw. k der Anzahl der ermittelten Werte einer bestimmten Metrik entsprechen. Auf diese Weise ist es möglich, den Wertebereich für einzelne Metriken bzw. die größtmögliche absolute Abweichung von einer Vorgabe zu bestimmen.

Der Gesamt-***Durchschnitt*** eines bestimmten Wertes in Bezug auf einen Probestrom ist dagegen das arithmetische Mittel über alle Einzel-Durchschnitte der Probedurchläufe. Als Gleichung:

$$\bar{D}^{Metrik}_{\text{Probestrom}} = \frac{1}{N}\sum_{j=1}^{N} \bar{D}^{Metrik}_{\text{Durchlauf } j}, \text{ mit } \bar{D}^{Metrik}_{\text{Durchlauf } j} = \frac{1}{k}\sum_{l=1}^{k} Wert_l$$

Diese Aufspaltung in einzelne Durchschnitte geschieht ohnehin, da zunächst jeder Probedurchlauf analysiert wird. Zudem ist die Teilung in einzelne Zwischenergebnisse gegenüber der Verkettung aller Einzelwerte aus den verschiedenen Probedurchläufen für die Berechnungsgeschwindigkeit des Gesamtwertes von Vorteil, da die Teilberechnungen parallel betrieben werden können.

Eine weiterer Durchschnittswert ist die MAD, welche die ***mittlere absolute Abweichung*** von der Emulationsvorgabe angibt und wie folgt berechnet wird:

$$\text{MAD} = \frac{1}{k}\sum_{l=1}^{k} |Wert_l - \bar{D}^{Metrik}_{Durchlauf}| \tag{5.3}$$

Zusätzlich zum Durchschnitt und zur MAD wird der Zentralwert, auch ***Median*** $x_{0.5}$ genannt, ermittelt. Bei einer geordneten Liste von Werten befindet sich der Median für eine ungerade Anzahl an Werten in der Mitte der Liste. Bei einer geraden Anzahl an Listenelementen entspricht $x_{0.5}$ dem Mittelwert aus dem Endwert der ersten Hälfte und dem Anfangswert der zweiten Hälfte. In Abbildung 5.13 ist ein Beispiel für diesen Fall zu sehen. Gleichzeitig demonstriert das Beispiel den Vorteil des Median gegenüber dem Durchschnitt. Ausreißer, wie in diesem Fall der Wert 300, haben keinen Einfluss auf den Median. Für die Betrachtung der Verteilung der Werte ist daher der Median vorzuziehen.

Weiterhin ist die Berechnung und Darstellung der ***Häufigkeitsverteilung*** wichtig. Mit dieser lassen sich neben dem Modalwert x_D, auch die Standardabweichung σ und damit die Streuung der emulierten Werte feststellen. Zusammen mit dem Durchschnitt und dem Median ermöglicht der Wert von x_D, außerdem, Aussagen über das *Schiefemaß* der Verteilung zu treffen. Dieses gibt einen Hinweis auf die Art der Verteilung. Beispielsweise ist eine Poissionverteilung *rechtsschief*, eine Normalverteilung symmetrisch und eine Binominalverteilung *linksschief*. Ein Beispiel dafür ist in Abbildung 5.14 zu sehen. Zur Vereinfachung sind bei diesem die einzelnen Verteilungen

$$\text{Liste} = [1,\ 2,\ 4,\ \overbrace{5}^{\text{erste Hälfte}},\ \overbrace{7,\ 8,\ 9,\ 300}^{\text{zweite Hälfte}}]$$

$$\text{Median} = \frac{1}{2}(5+7) = 6$$

$$\bar{D} = \frac{1}{8}\sum_{i=1}^{8}(\text{Liste[i]}) = 42$$

Abbildung 5.13: Beispiel für die Ermittlung des Median einer Liste mit gerader Anzahl an Elementen.

allgemein grafisch als Verteilungsfunktion (engl. *Probability Distribution Function* (PDF)) sowie als kumulierte Verteilungsfunktion (engl. *Cumulative Distribution Function* (CDF)) dargestellt.

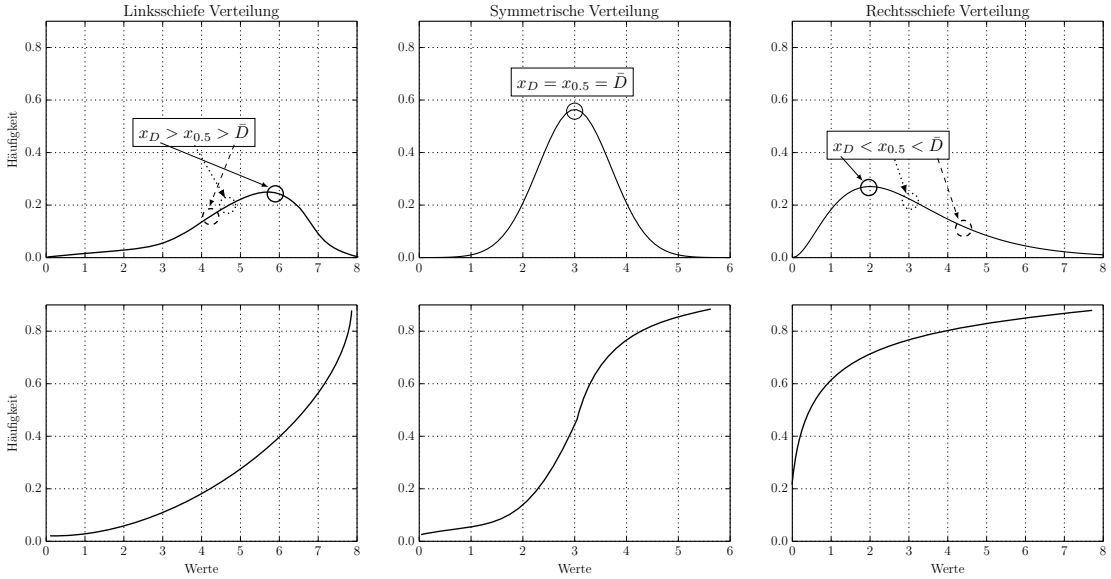


Abbildung 5.14: Schematischer Verlauf von links-, rechtsschiefen sowie symmetrischen Verteilungen. Oben: PDF ,Unten: CDF

Die Mehrzahl der vorgestellten Emulatoren besitzen Einstellungsmöglichkeiten bzgl. der Verteilungsart bzw. Wahrscheinlichkeit von z. B. Paketeffekten oder Verzögerungen. Für eine Gegenüberstellung der Gemessenen und der vorgegebenen Verteilung können daher die drei statistischen Lageparameter x_D, $x_{0.5}$ und der Durchschnitt verwendet werden. Die Zusammenhänge sind in Abbildung 5.14 in den Graphen der jeweiligen PDF zu sehen. Sind alle Lageparameter annähernd gleich groß, so besteht die Möglichkeit, dass die gemessenen Werte einer Normalverteilung unterliegen. Der Graph der CDF eignet sich, ergänzend zur PDF, besonders, um den Grad der Streuung bzw. Abweichung abzulesen. Je steiler der Graph ist, desto konzentrierter liegen Werte vor.

	ΔOWD^j-Liste in *ms*
Senderseite	2, 3, 2, 2, 3, 3, 2, 2, 2, 2
Empfängerseite	3, 2, 2, 3, 40, 1, 5, 2, 2, 3

Tabelle 5.2: Beispielwerte für die Betrachtung der Wertstabilität mit Hilfe von CoDVwBMA.

Der ***Variationskoeffizient*** (engl. *Coefficient of Variation* (CoV)) stellt eine Vergleichsmöglichkeit zwischen Versuchen mit unterschiedlichen Vorgaben, wie der Verzögerungsdauer, dar. Aber auch innerhalb eines Probedurchlaufes können mit Hilfe des CoV verschieden große Betrachtungsintervalle miteinander verglichen werden, um so z. B. die Wertstabilität oder Streuung innerhalb verschiedener Zeitspannen betrachten zu können. Der Variationskoeffizient ist das Verhältnis aus dem Durchschnitt und der Standardabweichung:

$$\text{CoV} = \frac{\sigma}{\overline{D}}$$

Die Abhängigkeit der Standardabweichung vom Durchschnitt wird dadurch aufgelöst. Messreihen, deren Probeströme höhere Mittelwerte für eine bestimmte Metrik annehmen können, als solche mit geringeren Werten, beinhalten entsprechend auch höhere Standardabweichungen. Beispielsweise entstehen höhere Abweichungen bei einer emulierten Verzögerung von 300 *ms* als bei einer Vorgabe von 1 *ms*. Die Wertebereiche der Messergebnisse unterscheiden sich deutlich.

Um die Wertstabilität einer Metrik innerhalb eines Paketstromes für verschiedener Zeitspannen untersuchen zu können, wird die Berechnung des in Kapitel 3 vorgestellten CoTV, angepasst. Ursprünglich verwenden die Autoren von [Sha+10] diesen als Maßeinheit für die Variation von Verzögerungen, indem die empfangene Datenrate mit der Gesendeten für unterschiedliche Zeitintervalle verglichen wird.

In dieser Arbeit werden neben der Datenrate ebenfalls direkt die Messwerte der OWD und der ΔOWD^j verwendet. Anders als in [MFA10] werden dabei für die Mittelwertbildungen nicht mehrere feste Intervalle, sondern jeweils ein einzelnes gleitendes Zeitintervall verwendet. Abhängigkeiten des Mittelwertes bezüglich der Intervallgrenzen werden so aufgelöst und die Gewichtung von vielen geringen Wertschwankungen gegenüber wenigen starken Schwankungen nimmt mit steigender Intervallänge ab. Das folgende Beispiel demonstriert das Vorgehen zur Berechnung des CoDVwBMA. Betrachtet wird dabei die Stabilität von gesendeten und empfangenen ΔOWD^j-Werten (vgl. Tabelle 5.2) innerhalb von sechs verschiedenen Zeitintervallen ΔT_j $(1 \leq j \leq 6)$. Zur besseren Übersicht wurden für das Beispiel lediglich zehn Messwerte gewählt und der Betrachtungsbereich eher grob gewählt. Üblichweise sind die Wertemengen deutlich größer und die Zeitintervalle beginnen bei 1 *ms*. Im Beispiel tritt auf der Empfängerseite eine hohe Abweichung mit dem Wert 40 *ms* und eine geringe Abweichung mit 5 *ms* auf.

In einem ersten Schritt wird für jedes Zeitintervall eine Liste mit den gleitenden Durchschnitten gebildet. Abbildung 5.15 veranschaulicht die Mittelwertbildung auf der Empfängerseite. Der erste Mittelwert des Zeitintervalls $\Delta T_1 = 10$ *ms* wird über die ersten vier Verzögerungsvariationen gebildet, da diese aufsummiert gleich dem Wert von ΔT_1 entsprechen und damit innerhalb dieses Zeitintervalls liegen. Das Zeitfenster wird für den zweiten Mittelwert um eine Stelle innerhalb der Liste verschoben. Der zweite Durchschnitt wird über den zweiten, dritten und vierten Messwert gebildet. Zwar liegt die Summe der einzelnen Messwerte unter dem Wert

von ΔT_1, jedoch würde die Einbeziehung des fünften Wertes dafür sorgen, dass die Summe außerhalb des Betrachtungszeitraumes liegt. Gleiches gilt für den dritten und vierten Mittelwert. Der Fortlauf des Zeitfensters blockiert solange vor der fünften Stelle bis diese Stelle selbst durch die Iteration des Algorithmus ausgewählt wird. Entsprechend ist der fünfte Mittelwert gleich dem fünften Messwert. Der sechste Mittelwert wird anschließend über die Messwerte sechs bis neun gebildet. Der siebente Durchschnitt dagegen wird aus dem siebenten bis neunten Messwert gebildet, da ein hinzufügen des zehnten Wertes eine Bereichsüberschreitung bedeutet. Weitere Mittelwerte sind nicht möglich, da das Zeitintervall von 10 *ms* nicht mehr erreicht werden kann. Aus diesem Grund entfallen ebenfalls alle Berechnungen nach dem fünften Mittelwert in den Mittelwertlisten $\bar{D}_{20}$, $\bar{D}_{30}$ und $\bar{D}_{40}$ (beide nicht in der Abbildung). Die Elemente der übrigen Listen $\bar{D}_{50}$ und $\bar{D}_{60}$ werden nach dem gleichen Vorgehen berechnet.

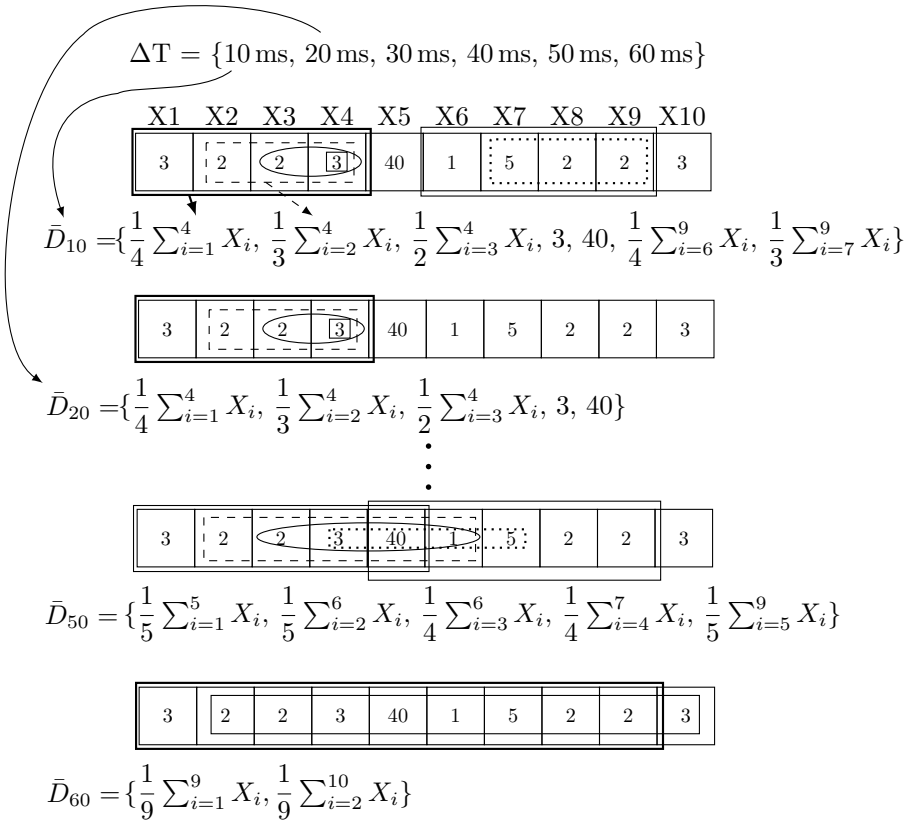

ΔT = {10 ms, 20 ms, 30 ms, 40 ms, 50 ms, 60 ms}
X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
3 2 2 3 40 1 5 2 2 3
$\bar{D}_{10} = \{\frac{1}{4}\sum_{i=1}^{4} X_i, \frac{1}{3}\sum_{i=2}^{4} X_i, \frac{1}{2}\sum_{i=3}^{4} X_i, 3, 40, \frac{1}{4}\sum_{i=6}^{9} X_i, \frac{1}{3}\sum_{i=7}^{9} X_i\}$
$\bar{D}_{20} = \{\frac{1}{4}\sum_{i=1}^{4} X_i, \frac{1}{3}\sum_{i=2}^{4} X_i, \frac{1}{2}\sum_{i=3}^{4} X_i, 3, 40\}$
$\bar{D}_{50} = \{\frac{1}{5}\sum_{i=1}^{5} X_i, \frac{1}{5}\sum_{i=2}^{6} X_i, \frac{1}{4}\sum_{i=3}^{6} X_i, \frac{1}{4}\sum_{i=4}^{7} X_i, \frac{1}{5}\sum_{i=5}^{9} X_i\}$
$\bar{D}_{60} = \{\frac{1}{9}\sum_{i=1}^{9} X_i, \frac{1}{9}\sum_{i=2}^{10} X_i\}$

Abbildung 5.15: Berechnung der Durchschnittlichen ΔOWD für das jeweilige Zeitintervall ΔT auf der Empfängerseite.

Die konkreten Ergebnisse der Mittelwertbildungen sind in Tabelle 5.3 zu sehen. Diese Werte bilden die Basis für den zweiten Schritt bei dem der CoDVwBMA für das jeweilig Zeitintervall gebildet wird. D. h. aus der Liste der Mittelwerte werden zum einen die Standardabweichung und zum anderen der Durchschnitt berechnet, um anschließend aus beiden das Verhältnis zu bilden. Als letzten Schritt werden die CoDVwBMA-Werte des Senders und des Empfängers im Diagramm in Abbildung 5.16 visualisiert. Erkennbar ist die deutliche Instabilität der Werte auf der

	Mittelwertlisten in *ms*	σ	Mittelwert($\bar{D}_j$)	CoDVwBMA
Senderseite	$\bar{D}_{10} = \{2.25, 2.5, 2.5, 2.5, 2.5, 2.25\}$	0.1178	2.4166	0.0487
	$\bar{D}_{20} = \{2.375, 2.375\}$	0.0	2.375	0.0
	$\bar{D}_{30} = \{2.29\}$	0.0	2.3	0.0
	$\bar{D}_{40} = \{2.29\}$	0.0	2.3	0.0
	$\bar{D}_{50} = \{2.29\}$	0.0	2.3	0.0
	$\bar{D}_{60} = \{2.29\}$	0.0	2.3	0.0
Empfängerseite	$\bar{D}_{10} = \{2.5, 2.\bar{3}, 2.5, 3, 40, 2.5, 3\}$	13.0759	7.9762	1.6394
	$\bar{D}_{20} = \{2.5, 2.\bar{3}, 2.5, 3, 40\}$	14.9683	10.0666	1.4869
	$\bar{D}_{30} = \{2.5, 2.\bar{3}, 2.5, 3, 40\}$	14.9683	10.0666	1.4869
	$\bar{D}_{40} = \{2.5, 2.\bar{3}, 2.5, 3, 40\}$	14.9683	10.0666	1.4869
	$\bar{D}_{50} = \{10, 9.6, 11.5, 12.25, 10\}$	1.0225	10.67	0.09583
	$\bar{D}_{60} = \{6.\bar{6}, 6.\bar{6}\}$	0.0	6.6666	0.0

Tabelle 5.3: Ergebnisse der Mittelwertberechnungen für die Betrachtung der Wertstabilität mit Hilfe von CoDVwBMA.

Empfängerseite für die Zeitintervalle bis 50 *ms*. Wählt man beispielsweise einen Betrachtungsbereich von 10 *ms*, so beträgt die Streuung um den Mittelwert $\sim 164\,\%$. Erst ab einem Intervall von 50 *ms* sinkt diese auf $\sim 10\,\%$. Anders verhält es sich auf der Senderseite. Hier sind für das erste Zeitintervall $\sim 5\,\%$ zu erkennen. Somit lässt sich für das Beispiel zusammenfassen, dass die Werte beim Sender für kleine Zeitbereich nur geringe Abweichungen vom Mittelwert aufweisen, während Empfängerseitig starke Streuungen zu verzeichnen sind und die verantwortlichen Ausreißer erst ab einem Bereich von 50 *ms* an Gewichtung verlieren.

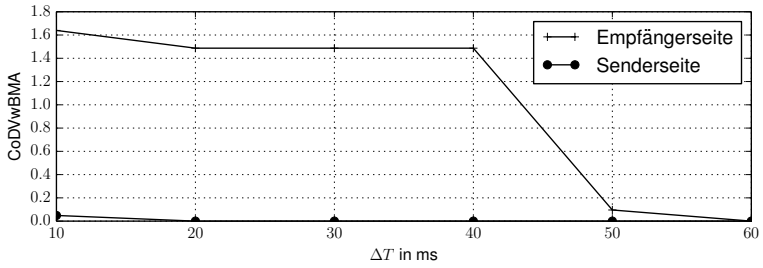


Abbildung 5.16: Visualisierung der berechneten CoDVwBMA für den Sender und Empfänger unter Berücksichtigung der Zeitintervalle $\Delta T = \{10\ ms, 20\ ms, \ldots, 60\ ms\}$.

5.3 Zusammenfassung

In Abbildung 5.17 sind die wesentlichen Bestandteile des hard- und software-technischen Aufbaus zusammengefasst. Die Abwägungen im ersten Teil dieses Kapitels ergeben einen Versuchsaufbau bei dem ein einzelner Rechner mit dem Emulator verbunden wird und die Messungen durchführt. Dazu ist lediglich eine zweite Netzwerkkarte notwendig, welche die Probeströme empfängt bzw.

sendet. Der Hauptvorteil ist dabei der Wegfall einer Uhrensynchronsation zwischen dem Sender und Empfänger.

Für den Software-technischen Aufbau können viele Komponenten aus einer vorgehenden Arbeit [Büs13], in welcher ein Ende-zu-Ende Messwerkzeug namens NORA entwickelt wurde, mit einigen Anpassungen übernommen werden. Dazu zählen hauptsächlich Konzepte zur Paketerfassung und Messwertanalyse. Fehlende Teile wie die Versuchssteuerung oder Datenhaltung werden durch Erweiterungen der ursprünglichen Software hinzugefügt. Zusätzlich bringen Verbesserungen am Paketgenerator stabilere Probeströme mit höheren Paketraten zustande.

Die Eigenschaften der Probeströme und die Vorgaben am Emulator bilden die Versuchsparameter. Ist der Emulator außerdem in der Lage Verteilungen oder Pfade nachzubilden, so erweitert sich die Parametermenge.

Allgemein wird in einem Versuch ein bestimmter Vorgabewert einer Metrik am Emulator eingestellt und mit bestimmten Paketströmen des Messsystems untersucht. Die Messwerte werden gespeichert und in der anschließenden Auswertung werden daraus die tatsächlich emulierten Werte berechnet. Mit statistischen Mitteln werden zudem die Höhe der Abweichungen und die jeweilige Stabilität einer Emulation analysiert. Dadurch lassen sich Aussagen über die Genauigkeit und die Leistungsfähigkeit des Emulators treffen.

Die konkreten Messungen unterteilen sich in Ermittlung der erreichbaren Wertgranularität und in die Untersuchung des möglichen Wertebereiches in Abhängigkeit von bestimmten Toleranzen. Für jede Vorgabe werden gleichermaßen mehrere Paketraten mit unterschiedlichen Paketgrößen verwendet. In Bezug auf die Anzahl der Pakete und Probedurchläufe unterscheiden sich jedoch die einzelnen Vorgaben, je nachdem ob Paketeffekte, Verzögerungen oder Datenraten untersucht werden. Die genauen Wertangaben der Versuchsparameter werden im nächsten Kapitel zur Umsetzung der beschriebenen Konzepte diskutiert und festgeschrieben.

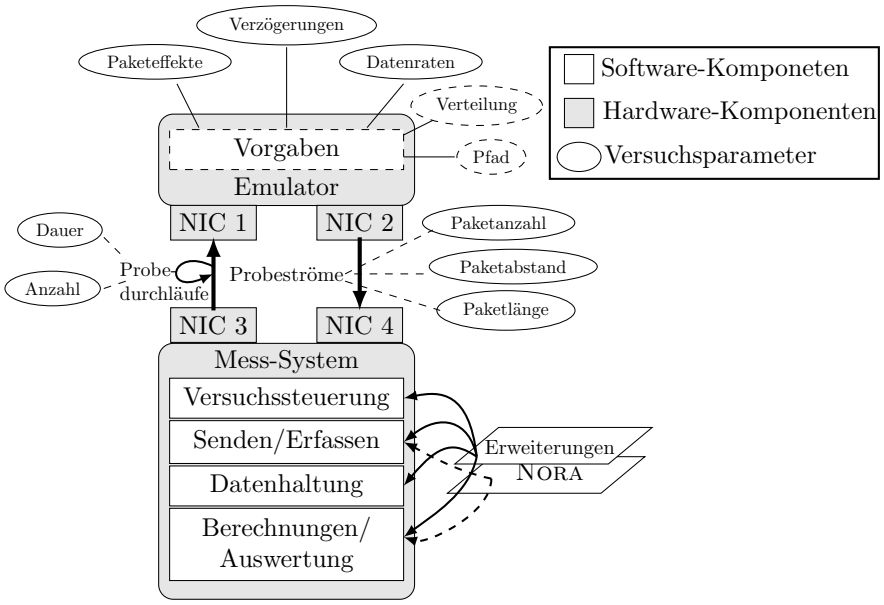


Abbildung 5.17: Überblick über den Hard- und Software-technischen Versuchsaufbau inklusive der Versuchskomponenten sowie -parameter.

6 Umsetzung

In diesem Kapitel werden die Umsetzungen der vorgestellten Konzepte beschrieben, wobei die Schwerpunkte dabei auf dem Versuchsaufbau und der konkreten Vorgehensweise bei den einzelnen Messungen liegen. Der Aufbau orientiert sich an dem des vorhergehenden Kapitels und beginnt mit der Versuchsanordnung.

6.1 Versuchsaufbau

6.1.1 Hardwaretechnischer Aufbau

Für die Messungen werden zwei handelsübliche Personal Computer mit jeweils zwei Netzwerkkarten verwendet. Die technischen Spezifikationen sind in Tabelle 6.1 aufgelistet. Der Messrechner enthält für die Erfassung der Messwerte die leistungsstärkeren Komponenten. So wird sichergestellt, dass der Messrechner eine höhere Paketrate verarbeiten kann als der Emulator und somit unerwünschte Verzögerungen oder Paketeffekte nicht bei diesem Rechner zu finden sind. Diese Ressourcenaufteilung bedeutet jedoch nicht, dass der Emulator in seiner Leistungsfähigkeit eingeschränkt wird. Viele der vorgestellten Emulatoren wurden auf ähnlichen oder schwächeren Systemen getestet (vgl. Tabelle 3.10). Außerdem bauen einige der Emulatoren auf Betriebssystemen auf, welche neuere Prozessoren, Bussysteme oder eine höhere Arbeitsspeicherzahl nicht unterstützen (vgl. ebenfalls Tabelle 3.10).

Als Netzwerkkarten werden „Intel Pro 1000“-Karten verwendet, wobei jeweils eine Karte über den PCI-Steckplatz und die andere über den „PCIe x1“-Steckplatz angeschlossen ist. Standardmäßig ist der PCI-Bus ein geteiltes Medium und hat eine theoretische Übertragungsrate von bis zu 2 $GBit/s$ (PCI 2.1 32-Bit) in beide Richtungen. In der Praxis liegt der Wert weit unter diesem theoretischen Maximum, da sich alle angeschlossenen PCI-Geräte und fest installierten „on-board“-Chips den PCI-Bus bidirektional teilen. Die Zuteilung des Busses geschieht über ein Zeitmultiplex-Verfahren. D. h. selbst wenn lediglich zwei Gigabit-Netzwerkkarten angeschlossen sind, würden sich beide Karten den Bus teilen und so einander beeinflussen. Um dieses zu vermeiden, wird die zweite Netzwerkkarte per PCIe-Bus angeschlossen. Dieser Bus ist unabhän-

	CPU	RAM	NIC
Messrechner	Intel i7 820 @2,93 GHz	8 GByte mit 1333 MHz DDR3	- Intel Pro GT 1000 (Tx) - Intel Pro PT 1000 (Rx)
Emulator	Intel Core 2 6600 @2,4 GHz	4 GByte mit 667 MHz DDR2	- Intel Pro GT 1000 (E1) - Intel Pro PT 1000 (E2)

Tabelle 6.1: Hardwarekonfiguration der Versuchsrechner (vgl. Abb. 5.3).

gig vom PCI-Bus oder anderen PCIe-Geräten und leitet die Daten ohne Multiplex-Verfahren, beispielsweise zur CPU oder zum Arbeitsspeicher, weiter.

Die Hauptplatine des Emulator-Rechners unterstützt den PCIe-Standard 1.1 und die „Intel Pro PT 1000“ wird wiederum über den „PCIe x1“-Steckplatz angeschlossen. Insgesamt ist daher exklusiv eine Datenübertragung vom maximal $4\ GBit/s$ in beide Richtungen möglich. Der Messrechner unterstützt den Standard 2.0, was bedeutet, dass die verwendete Netzwerkkarte exklusiv eine Transferrate von $8\ GBit/s$ nutzen kann. In beiden Fällen ist eine Nutzung von zwei PCIe-Karten ist nicht möglich, da jeweils nur ein Steckplatz auf der Hauptplatine verfügbar ist. D. h., die Grafikkarte des jeweiligen Rechners blockiert den anderen PCIe-Steckplatz.

Die beschriebene Aufteilung der Netzwerkkarten auf verschiedene Busse ist somit eine weitere Maßnahme, Engstellen und damit ungewollte Paketeffekte sowie Verzögerungen zu vermeiden.

Die Verbindung zwischen beiden Rechner ist ein weiterer Ansatzpunkt zur Vorbeuge von Engpässen. Beide Rechner werden direkt, ohne Zwischenverteiler wie einem Switch oder Hub, miteinander verbunden. Die verwendeten Netzwerkkabel erfüllen den „Cat.6“-Standard, sind doppelt geschirmt und haben eine Länge von $50\ cm$. Dadurch gewährleisten diese Kabel Datenraten von mehreren $GBit/s$.

Eine Fotografie des endgültigen Aufbaus aus Hardware-technischer Sicht ist in Abbildung 6.1 zu sehen.

6.1.2 Softwaretechnischer Aufbau

Die Pakete müssen für die Untersuchungen vom Messrechner aus über den Emulator zurück zum Ausgangspunkt versendet werden. Da auf dem Messrechner zwei Netzwerkkarten installiert sind, bedarf es einigen Vorkehrungen bzgl. der Netzwerkkonfigurationen, welche verhindern, dass der Emulator ausgelassen wird und stattdessen die Pakete direkt zwischen den beiden Netzwerkkarten ausgetauscht werden. Dazu werden die Karten logisch zwei unterschiedlichen Subnetzen zugeordnet. Dies geschieht mit Hilfe der Einstellung bzgl. der Subnetzmaske. Wichtig ist, dass der Emulator ebenfalls in beiden Subnetzen vertreten ist. Außerdem erhält die Karte, welche die Pakete versendet, eine anderes Gateway als die Empfängerkarte. In diesem Fall ist die IP-Adresse des jeweiligen Gateways die über das Netzwerkkabel verbundene Netzwerkkarte des Emulators. Die konkreten Adresseinstellungen und die verwendeten Subnetzmasken sind im Schema in Abbildung 6.2 aufgeführt.

Zum softwaretechnischen Versuchsaufbau gehören neben der Netzwerkkonfiguration, auch die Erstellung der Software-Komponenten zur Messung und Auswertung. Die Prozessorarchitektur des Messrechners erlaubt es, die Software-Komponenten des Versuchsaufbau in mehreren eigenständigen Prozessen zu gliedern. Dadurch ist es erst möglich die Paketgenerierung und -erfassung gleichzeitig durchzuführen. Der Hauptprozess ist dabei die Versuchssteuerung (vgl. Abb. 5.4). Diese startet und beendet alle anderen Prozesse. Für die Gewinnung der Messwerte sind dies der Paketgenerator und die sender- sowie empfängerseitige Paketerfassung. Für die Auswertungen wird ein Prozesspool angelegt, dessen Elemente parallel alle Messergebnisse der einzelnen Probeströme analysiert und die Werte der Metriken sowie Statistiken berechnet.

Versuchssteuerung Die Prozessgenerierung und -steuerung als solches wird mittels der Sprache Python umgesetzt. Die Prozesse selbst bzw. die Verarbeitungsroutinen sind je nach zeitlicher Anforderung ebenfalls in Python oder bei zeitkritischen Abläufen, wie z. B. der Paketgenerierung

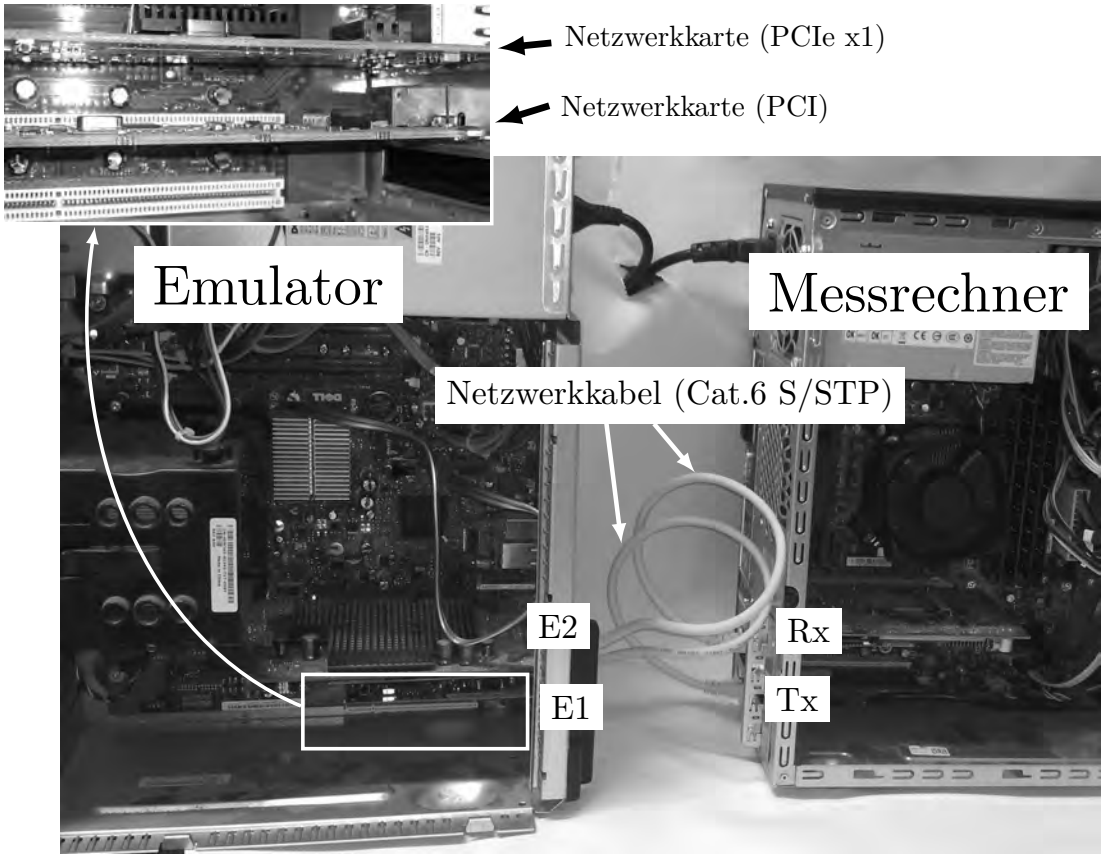


Abbildung 6.1: Fotografie des realen Versuchsaufbau. Oben: Vergrößerte Aufnahme der unterschiedlichen Netzwerkkarten.

und -erfassung, als Erweiterungsmodul in der Sprache C implementiert. Die Erweiterungsmodule werden wiederum von Python aus mit bestimmten Parametern aufgerufen und durch sogenannte „Callback"-Funktionen innerhalb des C-Programmes gesteuert. Ein Beispiel dazu ist in Abbildung 6.3 aufgeführt. In diesem startet der Steuerungsprozess über einen Funktionsaufruf mit Argumenten bzgl. der Netzwerkkartenidentität, Erfassungsfiltern, Zeitschranken, usw. den Prozess der Paketerfassung. Der dazugehörige Quelltext zeigt neben der Einbindung und dem Aufruf des Erweiterungsmoduls die „Callback"-Funktion, welche beispielsweise den Wert „2" zurückgibt, wenn der Sender alle Pakete versendet hat. Das Erweiterungsmodul ist in C geschrieben. Die darin enthaltenen Funktionen können, wie im Beispiel dargestellt, direkt in Python aufgerufen werden. Diese Vermischung der Sprachen verbindet die Geschwindigkeit der maschinennahen Sprache C mit der einfachen Prozesssteuerung durch Python.

Paketerfassung In Abbildung 6.3 ist außerdem ein Hinweis (grau hinterlegt) auf das genutzte Verfahren zur Paketerfassung zu erkennen. Die Pakete werden passiv mittels pcap-Treiber und der zugehörigen WinPcap-Bibliothek [13aa] erfasst und auch versendet. Die Erfassung der Pakete geschieht dadurch bereits nach deren Behandlung durch den Netzwerkkartentreiber. D. h.

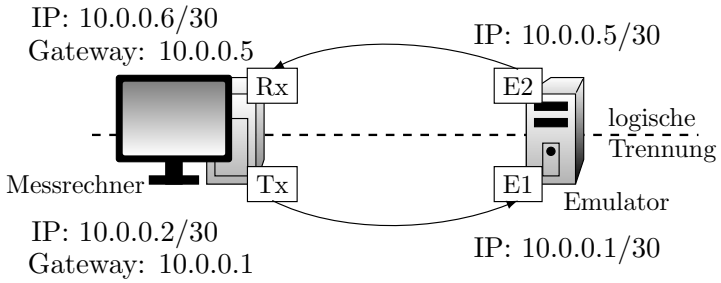


Abbildung 6.2: Netzwerkeinstellungen des Versuchsaufbaus.

die Zeitstempel werden unabhängig von der Verarbeitungsgeschwindigkeit einer Anwendung möglichst nahe am Übertragungsmedium vergeben.

Paketgenerator Die WinPcap-Bibliothek bietet neben der Paketerfassung auch die Möglichkeit Pakete zu versenden. Dabei werden Sendefunktionen innerhalb des Betriebssystemkerns genutzt und auf Sockets innerhalb der Anwendungsebene verzichtet. Der Nachteil dabei ist der höhere Aufwand bei der Paketgeneration, da die Pakete im Rohformat als Byte-Sequenz übergeben werden müssen. Dem Gegenüber werden dadurch Verarbeitungsschritte eingespart (vgl. Abschnitt 5.4–„Paketgenerator“) und es wird eine höhere Paketrate erreicht. Die Pakete werden zunächst mit einer Sequenznummer, der Nummer des Probedurchlaufes sowie des Probestroms versehen und mit zufälligen Symbolen bis zur vorgegebenen Paketgröße aufgefüllt. Jedes generierte Paket wird zusammen mit einem Sendezeitpunkt, ebenfalls mit Hilfe der WinPcap-Bibliothek, in einem Puffer gespeichert, welcher vor dem Sendevorgang in den Kernelbereich geladen wird. Während des Sendevorgangs werden die Sendezeitpunkte mittels aktivem Warten (engl. Busy Wait) durchgesetzt. Dieses sorgt für eine erhöhte Prozessorlast, welche jedoch insgesamt, wegen der Leistungsfähigkeit des Messrechners, nicht ins Gewicht fällt.

Analyse/Berechnung Die Messergebnisse werden nach jedem Probedurchlauf in einer neuen Datei im „csv“-Format gespeichert. Diese umfassen die Paketnummern und Zeitstempel der erfassten Pakete. Nach der Messung aller Probeströme eines Versuches beginnt die Analyse und Berechnung der Metrikwerte auf Grundlage dieser Messergebnisse. Die Werte der Metriken wiederum bilden die Basis für die Statistiken. Viele der verwendeten Implementationen zur Analyse entstammen dem Messwerkzeug Nora, welches ebenfalls zu Teilen in Python und C geschrieben wurde. So wird wie bei Nora, beispielsweise jeder Probedurchlauf vor der Analyse von Verzögerungen oder Datenraten auf Interrupt-Verschmelzungen hin untersucht und bei Bedarf entsprechend behandelt. Neu dagegen ist die Möglichkeit, aufgrund der Trennung von Messung und Auswertung, Berechnungen parallel durchzuführen.

Für die Berechnungen wird ein Prozesspool mit der Größe von acht Elementen angelegt, welches der Summe der verfügbaren physischen und virtuellen Prozessoren entspricht. Jedem Prozess innerhalb des Pools wird ein Probestrom zur Bearbeitung zugeordnet. Wenn ein Prozess mit den Berechnungen fertig ist, wird diesem der nächste unbearbeitete Probestrom zugeordnet. Die Ergebnisse der Berechnungen werden in weiteren „csv“-Dateien oder als Diagramme in „pdf“-Dateien gespeichert.

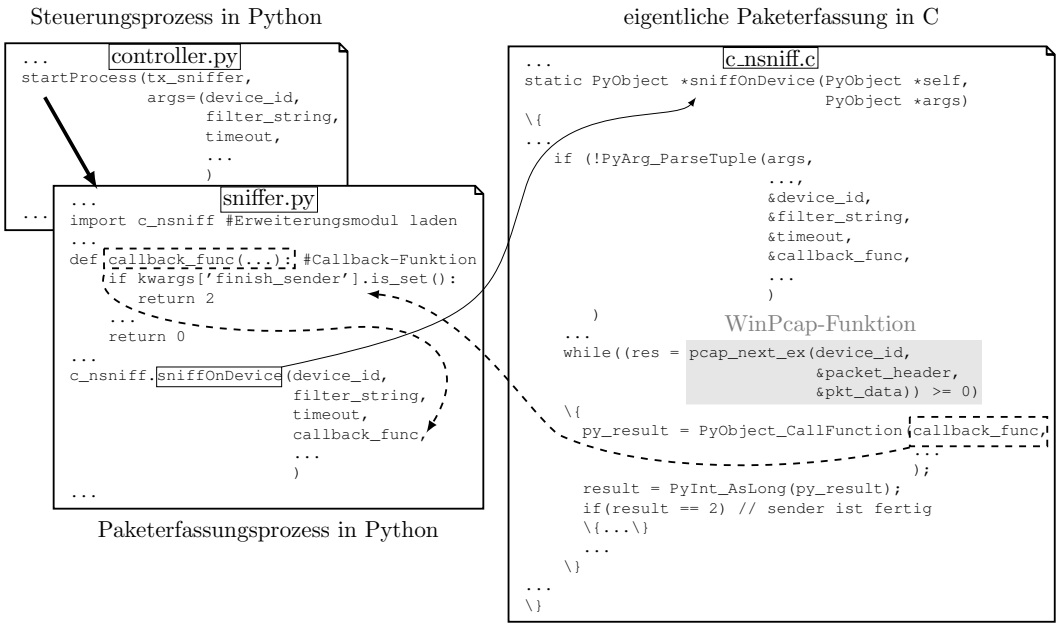


Abbildung 6.3: Beispiel für die Prozessgenerierung der Paketerfassung sowie das Zusammenwirken von Python und einem C-Erweiterungsmodul.

6.2 Vorgehensweise

An dieser Stelle werden die Beschreibungen der in Abschnitt 5.2 erläuterten ersten beiden Phasen um explizite Wertangaben bzgl. der Messparameter ergänzt. In allen Untersuchungen zur Granularität werden, sofern der Emulator dies unterstützt, jeweils die Pfadlängen $H = \{1, 10, 30\}$ verwendet. Die Pfadlänge $H_3 = 30$ orientiert sich an der maximalen Anzahl an messbaren *Hops* mittels des Tools TRACEROUTE.

Für die Ermittlungen des Wertebereiches wird ein einzelner Pfadabschnitt untersucht, da sich die Vorgabewerte in Bezug auf den Gesamtpfad im Falle der Paketeffekte und Verzögerungen erhöhen bzw. im Fall der Datenraten nicht verändern. Beispielsweise ergibt sich bei der Pfadlänge H_2, einer vorgegebenen Verlustrate von $10\,\%$ und ausgehend davon, dass alle Abschnitte die gleiche Verlustrate besitzen, eine Gesamtrate von:

$$\begin{aligned} \mathrm{PLR}_{gesamt} &= 1 - \prod_{j=1}^{H} (1 - \mathrm{PLR}_{j\text{-te Vorgabe}}) \\ &= 1 - (1 - 0,1)^{10} \\ &= 0,65 \text{ bzw. } 65\,\% \end{aligned}$$

Alle Messungen haben jedoch die verwendeten Stromeigenschaften gemeinsam. D. h., es werden die Paketgrößen $a = \{64, 512, 1024, 1500\}_{Byte}$ sowie die Paketabständen $b = \{20, 120, 1000\}_{\mu s}$ untersucht. Dadurch ergeben sich pro Pfadlänge insgesamt 12 Datenraten und entsprechend eine $3x4$ Ergebnismatrix mit 12 Einzelergebnissen für jede Wertangabe und Statistik. Tabelle

(b) \ (a)	(a1) 64 Byte	(a2) 512 Byte	(a3) 1024 Byte	(a4) 1500 Byte
(b1) 20 μs	(a1)(b1)	(a2)(b1)	(a3)(b1)	(a4)(b1)
(b2) 120 μs	(a1)(b2)	(a2)(b2)	(a3)(b2)	(a4)(b2)
(b3) 1000 μs	(a1)(b3)	(a2)(b3)	(a3)(b3)	(a4)(b3)

Tabelle 6.2: Allgemeine Ergebnismatrix und gleichzeitig Übersicht über die verwendeten Paketgrößen (a1)–(a4) und -abstände (b1)–(b3) (vgl. Abb. 5.9).

(b) \ (a)	(a1) 64 Byte	(a2) 512 Byte	(a3) 1024 Byte	(a4) 1500 Byte
$\mathrm{PPS}_{(b1)} = 50K$	26 *MBit*/*s*	205 *MBit*/*s*	410 *MBit*/*s*	480 *MBit*/*s*
$\mathrm{PPS}_{(b2)} = 8,3K$	4 *MBit*/*s*	34 *MBit*/*s*	68 *MBit*/*s*	100 *MBit*/*s*
$\mathrm{PPS}_{(b3)} = 1K$	0,5 *MBit*/*s*	4 *MBit*/*s*	8 *MBit*/*s*	12 *MBit*/*s*

Tabelle 6.3: Übersicht über die resultierenden Paket- sowie Datenraten bei der Kombination aus den Werten für (a1)–(a4) und (b1)–(b3) (vgl. Abb. 5.9).

6.2 veranschaulicht die verallgemeinerte Matrix. Die Kennzeichnungen der einzelnen Paketstromeigenschaften orientiert sich dabei am Abschnitt 5.2–„allgemeine Vorgehensweise“. Die Kombinationen aus Paketgröße und -abstand ergeben wiederum die Tabelle 6.3 aufgeführten Paket- und Datenraten.

Zusätzlich wird ein vierter Abstand (b4) für die Paketgröße (a2) gewählt, welcher dem Paketabstand der MLFFR des Versuchsaufbaus für diese Paketgröße entspricht. Die genauen Wertvorgabe für die minimalen Paketabstände kann daher erst nach Abschluss der Vorfelduntersuchungen angegeben werden. Der Grund für dieses Vorgehen wird im Folgenden erläutert.

Der Wert für (a1) entspricht der kleinsten erlaubten Ethernet Paketgröße und begründet sich auf der Erkennung von Paketkollisionen bei maximaler Netzausdehnung (IEEE 802.3 CSMA/CD (Ethernet)). Bei Gigabit Ethernet würde die sich die maximale Ausdehnung im Vergleich zu Netzwerken mit 10 *MBit*/*s* auf wenige Meter reduzieren. Daher wurde für Gigabit Ethernet die minimale Paketgröße auf (a2) = 512 *Byte* heraufgesetzt. Daher ergibt die Kombination des vierten Abstandes (b4) mit der Paketgröße (a3) die größte Paketrate für diesen Versuchsaufbau unter Beachtung der Kollisionserkennung. Der Wert für (a4) entspricht dem üblichen Standardwert für die *Maximum Transmission Unit* (MTU). Die Paketgröße von 1024 *Byte* für (a3) ist willkürlich gewählt.

Die Paketabstände starten individuell für jede Paketgröße mit dem kleinsten möglichen Wert für den Versuchsaufbau und damit mit dessen MLFFR für die jeweilige Paketgröße. Die folgenden Abstände dagegen werden für alle Größen gleich gesetzt, um so Vergleichsmöglichkeiten zu schaffen. Als Ausgangspunkt dient dabei die maximal erreichbare Datenrate innerhalb des Versuchsaufbaus. Somit wird sich bei der Wahl von (b1) an der größten Paketlänge orientiert.

Probemessungen abseits der Vorfelduntersuchungen haben gezeigt, dass für den Versuchsaufbau und bei einer Paketgröße von 1500 *Byte* die verlustfreie Datenrate maximal 672 *MBit*/*s* beträgt. Da jedoch die Senderate für diese Paketgröße $\sim$ 920 *MBit*/*s* beträgt, könnte der Grund

```
...
irq18: em2 uhci4 <-- gemeinsamer Interrupt, em2 entspricht "E1"
irq20: atapci0
...
cpu0: timer
irq257: em1  <-- "E2"
cpu1: timer
```

Abbildung 6.4: Vereinfachte Ausgabe des Befehl `vmstat -i` zur Analyse von gemeinsam genutzten Interrupts.

dafür zum einen bei der Hardware des Emulator-Rechners und zum anderen bei der Weiterleitungsstrategie des Betriebssystems zu finden sein. So teilen sich beispielsweise auf dem Rechner des Emulators die Netzwerkkarte „E1“ (siehe Abb. 6.2) und der USB-Controller den gleichen PCI-Bus (vgl. Abschnitt 6.1.1). Das führt unweigerlich zu einer Beeinflussung und senkt damit die Datenrate. Abbildung 6.4 veranschaulicht die Problematik und zeigt eine Übersicht der Interrupt-Aufteilung, welche mittels des Befehls `vmstat -i` (FreeBSD) erstellt wurde. Für den Abstand ergibt sich bei einer maximalen Datenrate von 672 $MBit/s$ der Wert:

$$1500\,Byte/672\,MBit/s = 18\,\mu s$$

Der Wert von (b1) wird daher so gewählt, dass dieser etwas oberhalb des minimalen Paketabstandes für diese Paketgröße liegt. Für (b2) wird ein Wert gewählt, sodass die resultierende Datenrate für (a4) einen Wert von 100 $MBit/s$ annimmt. Der letzte Abstand spiegelt eine Erkenntnis aus dem Kapitel 3 wieder. Darin wird eine Untersuchung zum Emulator KauNet vorgestellt, bei der geringe Datenraten dafür Sorge tragen, dass Systemeinflüsse minimiert werden und eventuelle Abweichungen nur durch den Emulator entstehen können.

6.2.1 Vorfelduntersuchungen

In den Vorfelduntersuchungen werden zunächst die jeweiligen MLFFR für die unterschiedlichen Paketgrößen ermittelt. Dazu wird nach und nach die Paketrate erhöht und auf ungewollte Paketverluste untersucht. Sobald ein Verlust auftritt, wird die Untersuchung gestoppt und die vorhergehende Rate als MLFFR gespeichert. Aus diesen Ergebnissen lässt sich gleichzeitig der Abstand (b4) berechnen.

Anschließend werden die durchschnittlichen Grundverzögerungen für alle vorgestellten Stromeigenschaften, einschließlich des zuvor ermittelten Wert für (b4), gemessen. Wie in den anschließenden Messungen zur OWD, werden dafür jeweils 10000 Pakete versendet und analysiert.

Für die Ermittlung der Puffergröße wird wie im Beispiel in Abschnitt 5.2.1 vorgegangen. D. h. am Emulator wird eine Datenrate von 1 $MBit/s$ eingestellt, welche wiederum mit einer höheren Senderate untersucht wird.

6.2.2 Messungen

Nach dem Abschluss der Voruntersuchungen beginnen die eigentlichen Untersuchungen mit Hilfe der aufgeführten Stromeigenschaften und je nach Gegenstand der jeweiligen Messung mit Hilfe einer bestimmten Vorgabe am Emulator, Paketanzahl, Anzahl an Probedurchläufen etc.. Die eigentlichen Messungen werden, wie in Abschnitt 5.2.2–„konkrete Durchführung“ beschrieben, unterschieden in die Ermittlung der Granularität und des Wertebereiches.

Paketeffekte–Granularität

Die Anzahl der zu sendenden Pakete, beginnend bei 100 Paketen, wird schrittweise um den Faktor 10 erhöht. Gleichzeitig wird die Vorgabe des jeweiligen Paketeffektes, beginnend bei 1 %, am Emulator entsprechend um den Faktor 10 verringert. Die Anzahl der Probedurchläufe wird pro Vorgabe, Pfadlänge und Stromeigenschaft auf 100 festgelegt. Die Erhöhung der Paketanzahl bzw. Verringerung des Vorgabewertes geschieht solange, bis alle Paketzahlen gemessen wurden, um so die Problematik mancher Pseudozufallsgeneratoren zu berücksichtigen. Die kleinste Effektrate wird anschließend ermittelt und ist erreicht, sobald die durchschnittliche relative Abweichung des jeweiligen gemessenen durchschnittlichen Paketeffektes von der Vorgabe größer als $\pm 3\,\%$ ist oder eine Paketstromlänge von hunderttausend Paketen erreicht wird. Die Granularität ist dann der Vorgabewert vor der Überschreitung der relativen Abweichung bzw. $0,001\,\%$ bei der Paketstromlänge von hunderttausend Paketen. Für die Untersuchung der Pfadlänge erhält jeder Abschnitt die gleiche Vorgabe. Dabei wird darauf geachtet, dass sich die Effektraten multiplizieren und sich somit die erwartete Rate erhöhen. Dennoch entspricht die kleinste Effektrate dem vorgegebenen Wert.

Paketeffekte–Wertebereich

Aus den Ergebnissen zur Granularität lassen sich die unteren Grenzen für die einzelnen Wertebereiche angeben. Um die oberen Grenzen zu erhalten werden neun Effektvorgaben gewählt. Die Vorgaben beginnen bei 1 % und werden schrittweise um 10 % erhöht. Hierbei werden je Vorgabe und Stromeigenschaft 100 Probedurchläufe mit einer Paketanzahl gemessen, welche sich nach der möglichen Granularität richtet, jedoch mindestens 10000 Pakete beträgt.

Das Ergebnis sind die Statistiken FA.1 sowie FA.3 bis FA.7 und $x_{0.5}$ zu jedem Paketeffekt, jeder Vorgabe und Stromeigenschaft.

Verzögerungen–Granularität

Zur Messung der erreichbaren Granularität der emulierbaren Verzögerungswerte werden je Stromeigenschaft, Vorgabe und Pfadlänge 1000 Pakete in einem einzelnen Probedurchlauf verwendet. Da der Takt des Kernels ausschlaggebend für die Auflösung ist, dient dieser Wert als Ausgangspunkt für die Untersuchungen. Da die Taktung bzw. die resultierende Verarbeitungsfrequenz eine eher theoretische Angabe ist, liegt der eigentliche Startwert um den Faktor 10 höher. Ist die Taktrate jedoch ungewiss, so wird mit einer Vorgabe von 10 ms begonnen. Der Wert entstammt der Mindestangabe für die OWD des Emulators WANem (vgl. Tab. 3.7).

Die Vorgaben werden solange um 1 *ms* und ab 1 *ms* um 100 μs verringert, bis die relative Abweichung der durchschnittlichen OWD größer als $\pm 10\,\%$ ist, ungewollte Paketeffekte auftreten oder

die vorgegebene Verzögerung kleiner oder gleich der Grundverzögerung des Versuchsaufbaus ist. Die Granularität ist für beide Fälle die vorhergehende OWD.

Jeder Pfadabschnitt erhält die gleiche Vorgabe und es wird während der Berechnungen die Tatsache berücksichtigt, dass sich Verzögerungen entlang des Gesamtpfades aufsummieren. Dennoch entspricht die kleinste Verzögerung dem Vorgabewert.

Verzögerungen–Wertebereich

Die obere Grenze des Untersuchungsbereiches wird für die Verzögerungen auf eine Sekunde festgelegt, da diese OWD für die Emulation einer Satellitenverbindung völlig ausreicht und eine einzelne Untersuchung mit 10000 Paketen bereits $\sim 2,7$ Stunden dauert. Gemessen werden die Verzögerungen mit den Vorgaben 15 ms, 100 ms und 1000 ms für jede Stromeigenschaft.

Als Ergebnis werden die Statistiken FA.1 bis FA.7 für jede Vorgabe und Stromeigenschaft angegeben. Zusätzlich werden eventuell auftretende ungewollte Paketeffekte aufgeführt.

Datenraten–Granularität

Die Vorgabewerte für die einzelnen Stromeigenschaften beginnen bei 1 MBit/s und werden solange schrittweise um 100 kBit/s reduziert bis die relativen Abweichungen der durchschnittlichen gemessenen Datenrate beim Empfänger größer als $\pm 3\,\%$ ist, ungewollte Paketeffekte auftreten oder die Vorgabe den Wert 200 kBit/s annimmt. Grundlage für die jeweilige Messung sind einzelne Probedurchläufe mit der Dauer von 10 s.

Im Gesamtpfad erhält immer nur ein Abschnitt die aktuelle Vorgabe, alle anderen haben keine Beschränkung bzw. einen Wert der größer als die MLFFR des Versuchsaufbaus ist. Dadurch wird berücksichtigt, dass der Pfadabschnitt mit dem minimalen Vorgabewert die Gesamtrate bestimmt (vgl. Abschnitt 2.4.1). Die Positionierung dieses Abschnittes erfolgt, vom Sender aus gesehen, am Anfang der hinteren Hälfte des Pfades.

Datenraten–Wertebereich

Der Untersuchungsbereich wird durch die MLFFR des Versuchsaufbaus beschränkt. D. h., auch wenn der Emulator höhere Datenraten unterstützt, kann dies nicht durch diesen Aufbau gemessen werden. Vielmehr werden Abweichungen und die Wertstabilität von einzelnen emulierten Datenratenbeschränkungen mit Hilfe der vorgestellten Stromeigenschaften untersucht. Die Vorgabewerte orientieren sich an der maximalen Datenrate von den bekannten Netzwerkzugangstechnologien HSDPA, WLAN (802.11g), ADSL (ANSI T1.413) und Kabel (DOCSIS 3.0), daher $7,2\,MBit/s$, $54\,MBit/s$, $61\,MBit/s$ und $160\,MBit/s$.

Das Ergebnis dieser Untersuchungen sind die Werte alle Statistiken FA.1 bis FA.7 für jede Vorgabe und Stromeigenschaft, wobei ungewollte Paketeffekte berücksichtigt werden.

6.3 Zusammenfassung

Der hier vorgestellte Versuchsaufbau enthält Hardware-Komponenten, welche das gleichzeitige Versenden und Erfassen von Probepaketen auf einem einzelnen Messrechner ermöglichen. Dadurch kann auf eine Uhrensynchronisation verzichtet werden. Der Rechner des Emulators

		Stromeigenschaften			Vorgaben am Emulator	
		Paketanzahl	Probe-durchläufe	Probe-dauer		Abbruch bei
Granularität	Paketeffekte	[100; 1000; …; 100000]	100	–	[1; 0,1; …; 0,0001] %	1. kein Abbruch 2. rel. Abw.: ±3 % 3. 0,0001 % emuliert
	Verzögerungen	1000	1	–	[10; 9,9; …; 0,1] ms	1. rel. Abw.: ±10 % 2. ungew. Paketeffekte 3. 0,1 ms emuliert
	Datenraten	10000	1	10 s	[1; 0,9; …; 0,1] MBit/s	1. rel. Abw.: ±10 % 2. ungew. Paketeffekte 3. 0,1 MBit/s emuliert
Wertebereich	Paketeffekte	10000 o. Granularität	100	–	[1; 11; 21; …; 91] %	
	Verzögerungen	1000	1	–	[15; 50; 100; 1000] ms	
	Datenraten	10000	1	10 s	[7,2; 54; 61; 160] MBit/s	

Tabelle 6.4: Übersicht der verwendeten Messparameter zur Ermittlung der Granularität und des Wertebereiches.

wiederum verfügt über Leistungsmerkmale, die zumeist über den geforderten Merkmalen liegen. Zur Optimierung der möglichen Datenrate werden jeweils eine PCI und eine PCIe-Netzwerkkarte verwendet. Dadurch agieren die Karten auf unterschiedlichen Bussen und beeinflussen einander nicht. Dennoch könnten weitere Vorkehrungen getroffen werden, um die mögliche Datenrate noch weiter zu steigern. Dazu zählt beispielsweise die Verwendung von Hauptplatinen, welche noch einen weiteren PCIe-Anschluss haben, damit beide Netzwerkkarten über dieses Bussystem betrieben werden können.

Auch softwaretechnisch werden die Mehrprozessorarchitekturen dazu benutzt, zum einen parallel Messungen und zum anderen parallel Auswertungen durchzuführen. Die verwendeten Algorithmen sowie deren Implementationen werden teilweise dem Messtool Nora entnommen, angepasst und sogar verbessert. So ist beispielsweise der Paketgenerator in der Lage, stabilere und höhere Paketraten zu versenden. Fehlende Implementationen, wie z. B. die Auswertung von Variationskoeffizienten oder die Visualisierung der Ergebnisse in Form von Diagrammen werden als Erweiterung zugefügt.

Im zweiten Teil dieses Kapitels werden die Abfolge und die genauen Zahlenwerte der Vorgaben am Emulator sowie der verwendeten Paketgrößen, -abstände, -anzahl und Probedauer für die jeweiligen Untersuchung der Granularität bzw. des Wertebereiches erläutert. Eine Zusammenfassung dazu ist in Tabelle 6.4 zu finden.

Bevor jedoch mit den Messungen begonnen werden kann, müssen zunächst Funktionsnachweise für Teile der vorgestellten Umsetzung erbracht werden. Dazu zählt beispielsweise der richtige Routenverlauf der Probepakete. Die Validierung des vorgestellten Messkonzeptes ist das Thema des nächsten Kapitels.

7 Validation

Die vorgestellten Konzepte bzw. Umsetzungen zur Genauigkeits- und Leistungsanalyse müssen vor dem Einsatz auf deren korrekte Funktionsweise geprüft werden. Im Kapitel zur Anforderungsanalyse werden dazu die vier Nachweise QN.1 bis QN.4 erläutert, welche sicherstellen sollen, dass die Probepakete mit der eingestellten Größe sowie Rate ausschließlich vom Emulator beeinflusst und vom Empfänger erfasst werden. Fehlkonfigurationen sollen dabei vom System erkannt und angezeigt werden.

Dieses Kapitels gliedert sich entsprechend und beginnt mit dem Nachweise über die korrekte Funktion des Versuchsaufbaus bzw. des Netzwerkkonfiguration (QN.1). Anschließend werden die beiden Software-Komponenten untersucht, welche für die Generierung sowie Erfassung der Messdaten sorgen und damit grundlegend für alle Auswertungen sind (QN.2 und QN.3). Abschließend werden absichtlich mögliche Fehler in der Konfiguration bzgl. der Netzwerkkonfiguration und Messparameter ausgelöst und die Reaktion des Systems aufgezeigt.

7.1 Versuchsaufbau

Durchführung Der Versuchsaufbau wird jeweils mit fünf PING-Anfragen (ICMP-Requests) zwischen dem Sender (Tx):

1. und der ersten Netzwerkkarte des Emulators (E1)
2. sowie dessen zweiter Netzwerkkarte (E2)
3. und abschließend der Empfängerkarte (Rx)

getestet (vgl. Abb. 6.2). Für die korrekte Funktion, müssen E1, E2 und Rx auf die Anfragen mit jeweils fünf PING-Antworten (ICMP-Reply) reagieren. Die Nachweise werden zum einen durch die Ausgabe des Programmes PING und WIRESHARK erbracht.

Ergebnisse Die Ausgaben der beiden Programme sind zum einen in Anhang A.1 und zum anderen in Tabelle 7.1 zu finden. Letztere fasst alle drei Untersuchungen in einer Tabelle zusammen und zeigt den geforderten Ablauf von Anfrage und Antwort jeder einzelnen Station zwischen dem Sender und Empfänger. Alle Pakete wurden von jeder Netzwerkkarte beantwortet. Daher ist der Versuchsaufbau funktionsfähig.

7.2 Paketgenerierung und -erfassung

Durchführung Um die korrekte Funktion der Paketgenerierung für die anschließenden Messungen zu zeigen, werden Paketströme mit denjenigen Eigenschaften erzeugt und versendet, welche auch in den eigentlichen Untersuchungen zum Einsatz kommen. Im Detail ergeben sich daher die Paketstromeigenschaften wie sie in Tabelle 6.3 zu finden sind. Die Paketanzahl

No	Quelle	Ziel	Info
1	10.0.0.2	10.0.0.1	Echo (ping) request seq=121/30976
2	10.0.0.1	10.0.0.2	Echo (ping) reply seq=121/30976
3	10.0.0.2	10.0.0.1	Echo (ping) request seq=122/31232
4	10.0.0.1	10.0.0.2	Echo (ping) reply seq=122/31232
5	10.0.0.2	10.0.0.1	Echo (ping) request seq=123/31488
6	10.0.0.1	10.0.0.2	Echo (ping) reply seq=123/31488
7	10.0.0.2	10.0.0.1	Echo (ping) request seq=124/31744
8	10.0.0.1	10.0.0.2	Echo (ping) reply seq=124/31744
9	10.0.0.2	10.0.0.1	Echo (ping) request seq=125/32000
10	10.0.0.1	10.0.0.2	Echo (ping) reply seq=125/32000
11	10.0.0.2	10.0.0.5	Echo (ping) request seq=126/32256
12	10.0.0.5	10.0.0.2	Echo (ping) reply seq=126/32256
13	10.0.0.2	10.0.0.5	Echo (ping) request seq=127/32512
14	10.0.0.5	10.0.0.2	Echo (ping) reply seq=127/32512
15	10.0.0.2	10.0.0.5	Echo (ping) request seq=128/32768
16	10.0.0.5	10.0.0.2	Echo (ping) reply seq=128/32768
17	10.0.0.2	10.0.0.5	Echo (ping) request seq=129/33024
18	10.0.0.5	10.0.0.2	Echo (ping) reply seq=129/33024
19	10.0.0.2	10.0.0.5	Echo (ping) request seq=130/33280
20	10.0.0.5	10.0.0.2	Echo (ping) reply seq=130/33280
21	10.0.0.2	10.0.0.6	Echo (ping) request seq=131/33536
22	10.0.0.6	10.0.0.2	Echo (ping) reply seq=131/33536
23	10.0.0.2	10.0.0.6	Echo (ping) request seq=132/33792
24	10.0.0.6	10.0.0.2	Echo (ping) reply seq=132/33792
25	10.0.0.2	10.0.0.6	Echo (ping) request seq=133/34048
26	10.0.0.6	10.0.0.2	Echo (ping) reply seq=133/34048
27	10.0.0.2	10.0.0.6	Echo (ping) request seq=134/34304
28	10.0.0.6	10.0.0.2	Echo (ping) reply seq=134/34304
29	10.0.0.2	10.0.0.6	Echo (ping) request seq=135/34560
30	10.0.0.6	10.0.0.2	Echo (ping) reply seq=135/34560

Tabelle 7.1: Ausgabe von Wireshark für den Nachweis der Funktionalität des Versuchsaufbaus.

wird dabei auf 10000 festgelegt. Ein abschließender Test mit aufsteigenden Datenraten soll die Belastungsgrenze des Paketgenerators für jede Stromeigenschaft aufzeigen.

Gleichzeit zum Funktionstest der Paketgenerierung wird die Erfassung der ausgehenden und eingehenden Pakete geprüft. Dazu wird das Netzwerktool Wireshark eingesetzt. Gemessen werden mit diesem die Paketgröße und die einzelnen Paketabstände der aus- sowie eingehenden Pakete. Abschließend werden die Ergebnisse aus den Messungen mit Wireshark mit denen der implementierten Paketerfassung verglichen.

Ergebnisse Die Messergebnisse sind in Abbildung 7.1 als einfache Boxplots dargestellt. Die *Antennen* stellen in diesem Fall den Minimal- und Maximalwert der erfassten Paketabstände dar. In den Diagrammen der Senderseite lässt sich zum einen die Genauigkeit des Paketgenerators bei der Umsetzung der Vorgaben und gleichzeitig die Zuverlässigkeit der Paketerfassung erkennen. Als Referenz dienen die Ergebnisse, welche zur gleichen Zeit mittels Wireshark gemessen wurden. Die mittlere relative Abweichung der generierten Abstände in Bezug auf die Vorgabe

beträgt maximal -0,006 %. Außerdem arbeitet der Generator in nahezu allen Fällen stabil. Hierbei beträgt die relative Abweichung der einzelnen Extremwerte von der Vorgabe maximal -20 % (vgl. Abb. 7.1a).

Die implementierten Paketerfassungen am Sender und Empfänger liefern, bis auf wenige Ausnahmen, die gleichen Ergebnisse wie Wireshark. Die Ausnahmen sind in Abbildung 7.1a und 7.1d zu finden. Ein Grund dafür ist beispielsweise die angewendete Rundung der Werte bei Wireshark. Gerade bei Zeitwerten im Mikrosekundenbereich fällt das Aufrunden besonders ins Gewicht.

Somit lässt sich zusammenfassen, dass die Implementation des Paketgenerators stabil sowie zuverlässig arbeitet und die Werte der eigenen Paketerfassungen nahezu exakt mit denen von Wireshark übereinstimmen. Letzteres beruht auch auf der Tatsache, dass Wireshark und die eigene Erfassungen auf der gleichen Quelle (WinPcap) aufbauen.

7.3 Fehlererkennung

Durchführung Das Messsystem ist in der Lage vor dem Beginn einer Messung Fehler in der Konfiguration der Netzwerkeinstellung und bei der Eingabe der Messparameter zu erkennen. Dadurch soll der Bereich von möglichen Messfehlern weiter eingeschränkt werden.

Der Nachweis dafür wird durch absichtliche Fehlkonfigurationen erbracht. D. h. zum einen werden absichtlich Angaben zur Netzwerk- oder Versuchskonfiguration weggelassen und zum anderen werden fehlerhafte Angaben an das System übergeben.

Nachweis 1: fehlende Angaben Für das Messsystem ist u. a. die IP-Adresse des Empfängers wichtig. Fehlt diese Angabe in der Konfigurationsdatei für die Netzwerkeinstellungen erscheint eine Fehlermeldung mit dem Hinweis auf die fehlende Adresse. Der Inhalt einer entsprechenden Datei ist in Anhang A.2 zu sehen. Dort wurde in Zeile 20 die Angabe zur IP-Adresse auskommentiert. Beim Start der Messungen erscheint daraufhin:

```
Error: missing Argument(s): ['rx_ip_addr']
```

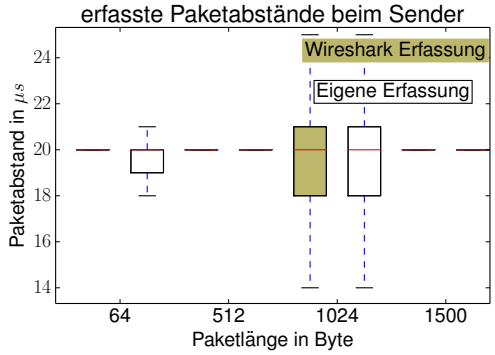
Das gleiche Verhalten zeigt sich bei fehlenden Eingaben bzgl. der Messungsparameter. Beispielsweise fehlen in der Konfigurationsdatei der Messung (siehe Anhang A.3) die Angaben zu den Paketabständen für die Messung „2“ (Zeile 26), entsprechend gibt das System die folgende Meldung aus:

```
ValueError: Given config for measurement '2' has not the
right number of values.
```

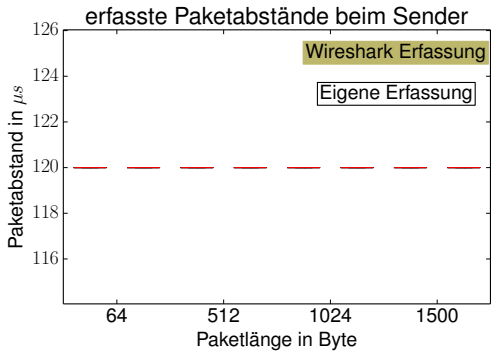
In der darauffolgenden Zeile ist die vollständige Konfiguration zu sehen.

Nachweis 2: fehlerhafte Angaben Neben diesen eher offensichtlichen Fehlern, sind es oft die fehlerhaften Konfigurationen, welche zu falschen Messwerten führen. Als Beispiel wird die MAC-Adresse soweit manipuliert, dass diese keine gültige MAC-Adresse mehr darstellt. Das System meldet daraufhin:

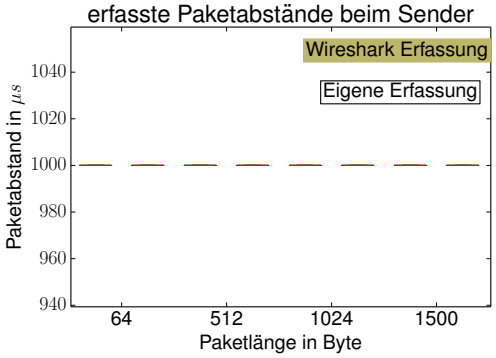
```
ValueError: Given Value is not a valid MAC-Address.
```



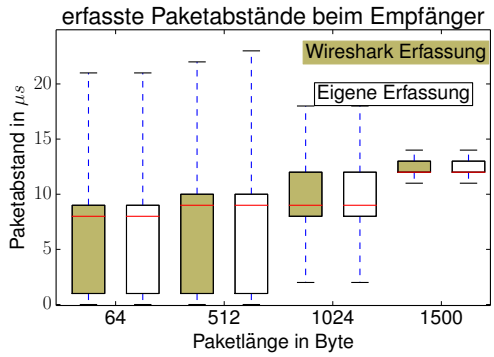
(a) vorgegebener Paketabstand: 20 μs



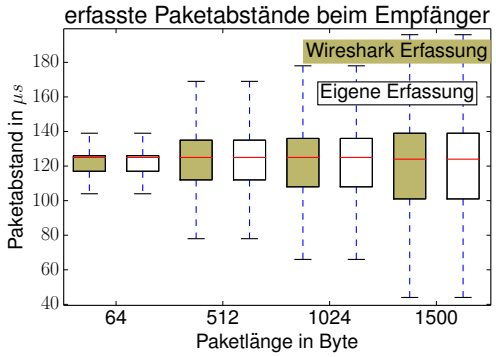
(b) vorgegebener Paketabstand: 120 μs



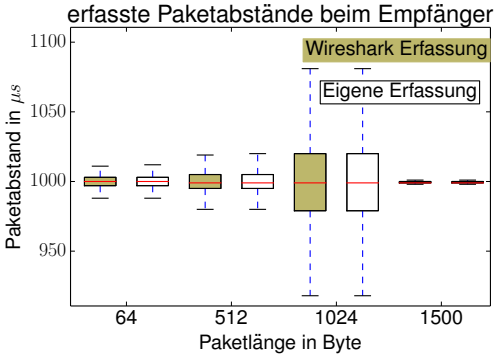
(c) vorgegebener Paketabstand: 1000 μs



(d) vorgegebener Paketabstand: 20 μs



(e) vorgegebener Paketabstand: 120 μs



(f) vorgegebener Paketabstand: 1000 μs

Abbildung 7.1: Messergebnisse der erfassten Paketabstände durch WIRESHARK und der eigenen Implementation auf der Senderseite (a)–(c) sowie Empfängerseite (d)–(e) als Boxplot für die verschiedenen vorgegebenen Paketabstände und -größen.

Gleiches gilt auch für eine falsche Netzwerkkarten-ID in Zeile 19 (Anhang A.2):

```
Error: given rx_device_id not found on this system.
```

Eine weitere Fehlerquelle sind die Vorgaben innerhalb der Konfigurationsdatei für die Messparameter. Beispielsweise ist in Anhang A.3 die minimale PPS größer als die maximale. Bei der Initialisierung des Messsystems erscheint eine entsprechende Fehlermeldung:

```
ValueError: maximum pps < minimum pps, or step width < 1
```

Ein Probedurchlauf muss eine Mindestdauer einhalten, um Überlappungen in der Paketerfassung zu vermeiden. Zur Sicherheit wird daher eine Dauer von mindestens einer Sekunde gefordert. Wird dennoch versehentlich ein geringerer Wert gewählt, muss das System dies erkennen. Für den Nachweis wurde ein Mindestdauer von 100 Millisekunden eingestellt. Daraufhin erscheint die Fehlermeldung:

```
ValueError: The minimum duration is 1 sec.
```

Neben den vorgestellten Beispielen, erkennt das Messsystem noch weitere Fehler. Dadurch werden bereits im Vorfeld Messfehler weitestgehend vermieden.

7.4 Zusammenfassung

In diesem Kapitel wird die Funktionsfähigkeit des Versuchsaufbaus, Paketgenerators sowie der Paketerfassungen gezeigt. Der Aufbau wird dabei mittels PING-Anfragen und -Antworten etappenweise vom Sender über die beiden Netzwerkkarten des Emulators zum Empfänger kontrolliert.

Dem Paketgenerator werden mehrere Vorgaben übergeben und die daraus resultierenden Paketströme werden mit Hilfe des Messtools WIRESHARK analysiert. Die Ergebnisse zeigen, dass der Generator zuverlässig und sehr genau arbeitet. Auch die Paketerfassungen auf der Sender- und Empfängerseite werden mit Aufnahmen von WIRESHARK verglichen. Auch hierbei zeigen die Ergebnisse, dass die Paketerfassungen zuverlässig arbeiten.

Im letzten Abschnitt werden absichtlich einige Fehler in die Konfiguration der Netzwerkeinstellung und der Messparameter eingebracht, um die Fehlererkennung des Messsystems vorzustellen. Diese Erkennung soll den Grad der Zuverlässigkeit des Versuchsaufbaus steigern, indem es bereits vor einer Messung mögliche Falschmessungen verhindert.

8 Durchführung

Im Rahmen dieser Arbeit werden mit Hilfe des vorgestellten Messkonzepts die Nachbildungen der Netzwerkmetriken *C*, OWD, PLR, PRR und PDR durch die vier Emulatoren DUMMYNET, KAUNET, NETEM und LINKTROPY analysiert. Jeder dieser Netzwerkemulatoren decken mehrere oder sogar alle der zu untersuchenden Metriken ab. Tabelle 8.1 bietet dazu einen Übersicht an (vgl. auch Tab. 3.10). Darüber hinaus sind DUMMYNET und KAUNET in der Lage mehrere Pfadabschnitte zu emulieren.

Weiterhin können nur ausgewählte Ergebnisse präsentiert werden, da die Gesamtanzahl an Messergebnisse außerhalb des Rahmens dieser Arbeit liegt. Die restlichen Auswertungen und auch die einzelnen Messdaten befinden sich auf beigelegten Datenträgern.

8.1 Vorfelduntersuchungen

Am Beispiel von KAUNET wird die allgemeine Durchführung zur Ermittlung der Grundverzögerung, MLFFR sowie Puffergröße erläutert. Die Vorfelduntersuchungen der übrigen Emulatoren verlaufen analog. Daher werden deren Ergebnisse abschließend zusammengefasst aufgeführt.

8.1.1 Grundverzögerungen

Die Messung der Verzögerung des Versuchsaufbaus wird ohne jedwede Emulationsregel durchgeführt. Speziell für KAUNET bzw. FREEBSD 7.3 ergeben sich die in Tabelle 8.2 aufgeführten Mittelwerte in Bezug auf die jeweilige Stromeigenschaft. Zur weiteren Veranschaulichung der Messergebnisse befinden sich in Abbildung 8.1 die entsprechenden Verteilungen als Box-plot mit den Mittelwerten (Sterne), Minima und Maxima (Antennen) sowie dem jeweiligen Medianen (mittlere Linie innerhalb der Box).

Erwartungsgemäß steigt die Verzögerung für wachsende Paketlängen. Die Unterschiede bleiben auch für sinkende Paketraten annähernd gleich. Zusätzlich ist zu erkennen, dass die Grundverzögerungen für geringere Paketraten höher ausfallen als bei den größeren Raten. Der

	C	OWD	PLR	PRR	PDR
LINKTROPY	X	X	X	X	X
DUMMYNET	X	X	X		
KAUNET	X	X	X	X	
NETEM		X	X	X	X

Tabelle 8.1: Übersicht über die Funktionsmöglichkeiten der ausgewählten Emulatoren in Bezug auf die zu untersuchenden Netzwerkmetriken.

		Paketgröße in Byte			
		64	**512**	**1024**	**1500**
Paket-abstand in μs	**20**	91	103	133	161
	120	111	119	146	172
	1000	148	157	185	208

Tabelle 8.2: Durchschnittliche Grundverzögerungen (in μs) des Versuchsaufbau für die jeweilige Paketstromeigenschaft unter FreeBSD 7.3 (KauNet).

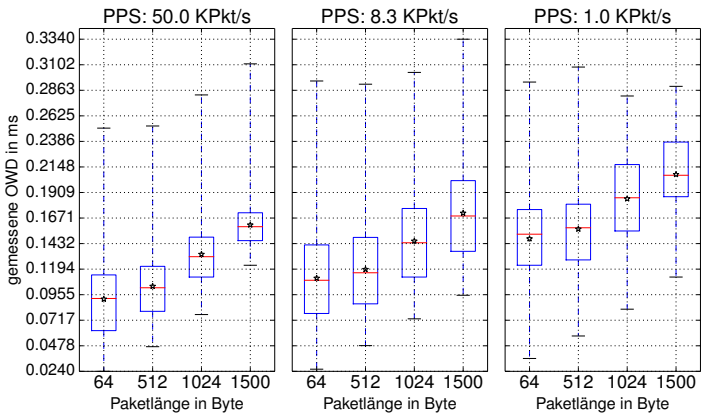


Abbildung 8.1: Box-Diagramme für die gemessenen Grundverzögerungen unter Verwendung der verschiedenen Stromeigenschaften am Beispiel von FreeBSD 7.3 (KauNet).

Grund dafür könnte in der Weiterleitungsstrategie von FreeBSD 7.3 zu finden sein, welche ein Bucket-System nutzt. D. h. bei einer sehr geringen Paketrate „füllen“ sich die Puffer nicht schnell genug, wodurch zusätzliche Verzögerungen entstehen.

Das gleiche Verhalten zeigt sich bei den anderen Emulatoren (vgl. Anhang A.4). Auch hier bedeutet ein Abfallen der Paketrate, ein Ansteigen der Grundverzögerung. In Tabelle 8.3 sind die gemittelten Grundverzögerungen für jeden Emulator zusammengefasst. Diese Mittelwerte werden in den eigentlichen Messungen berücksichtigt.

8.1.2 maximale verlustfreie Paketrate

Im Fall von KauNet bzw. FreeBSD 7.3 werden unabhängig von der Paketrate keine Pakete vom Betriebssystem verworfen, sondern stattdessen stark verzögert. In Abbildung 8.2 sind die Box-Diagramme der Verzögerungen für die verschiedenen Paketraten zu sehen. Es zeigt sich, dass bei einer Rate von 500 KPkt/s der zeitlicher Abstand über die vorhandene Grundverzögerung hinaus wächst. Eine Betrachtung der empfangenen und gesendeten Paketrate bei einer voreingestellte Rate von 500 KPkt/s zeigt, dass die Senderate vom System auf $\sim$ 126 KPkt/s reduziert wird. Somit wird für KauNet eine MLFFR von 125 KPkt/s angenommen und der minimale Paketabstand auf 8 μs festgelegt. Die Betriebssysteme FreeBSD 10.0 und Linux 3.2, auf denen jeweils Dummynet und NetEm basieren, verhalten sich gleichermaßen und puffern die Pakete. Die Zahlenwerte für die Dauer der Pufferung sind ähnlich denen von FreeBSD 7.3, daher werden auch hierbei

Paketabstand in μs			Paketgröße in Byte 64	512	1024	1500
	20	Linktropy	34	55	77	96
		FreeBSD 10.0 (Dummynet)	116	106	142	167
		FreeBSD 7.3 (KauNet)	91	103	133	161
		Linux 3.2 (NetEm)	52	75	110	177
	120	Linktropy	50	63	85	100
		FreeBSD 10.0 (Dummynet)	104	129	145	161
		FreeBSD 7.3 (KauNet)	111	119	146	172
		Linux 3.2 (NetEm)	61	83	102	127
	1000	Linktropy	51	100	111	138
		FreeBSD 10.0 (Dummynet)	161	153	191	204
		FreeBSD 7.3 (KauNet)	148	157	185	208
		Linux 3.2 (NetEm)	88	94	114	148

Tabelle 8.3: Mittelwerte der erfassten Grundverzögerungen (in μs) für den jeweiligen Emulator unter Verwendung von verschiedenen Paketgrößen und -abständen.

		Paketgröße in Byte **64**	**1024**	**64**	**1024**
Paketabstand	**20**	19.51 ms	390.25 ms	1285408 Bit/s	1029950 Bit/s
in μs	**120**	19.75 ms	385.31 ms	1280090 Bit/s	1019509 Bit/s

Tabelle 8.4: Durchschnittliche Verzögerungen (links) und Datenraten (rechts) für die jeweiligen Paketstromeigenschaften unter Verwendung von KauNet mit einer Datenratenbeschränkung von 1 MBit/s.

eine MLFFR von 125 KPkt/s angenommen. Die Ausnahme bildet der Hardware-Emulator, bei welchem auch bei einer Rate von 500 KPkt/s kein Ansteigen der Verzögerungen bzw. Paketverluste zu erkennen ist. Der Messrechner kann keine Paketrate in der Höhe generieren, bei welcher Verzögerungen oder Paketverluste am Linktropy auftreten. Eine MLFFR ist daher nicht messbar.

8.1.3 Puffergröße

Am Emulator wird eine Einstellung von 1 MBit/s als Datenratenbeschränkung gewählt. Untersucht werden dabei die Paketgrößen (a1) und (a3) sowie die Paketabstände (b1) und (b2). Daraus ergeben sich vier Vergleichswerte, welche alle auf die gleiche Puffergröße zurückzuführen sind. Für KauNet sind die gemittelten Werte der OWD und die Datenraten des Versuches in der Tabelle 8.4 zu sehen. Mit Hilfe von Gleichung 3.3 und diesen Werten ergeben sich die berechneten Puffergrößen in Tabelle 8.5. Somit lässt sich insgesamt feststellen, dass der Puffer von KauNet standardmäßig ∼50 Pakete zwischenspeichern kann.

Eine Gesamtübersicht der Puffergrößen der verwendeter Emulatoren, außer NetEm, ist in Tabelle 8.6 aufgeführt. Es sei dabei angemerkt, dass sich diese Größen einstellen lassen und die gezeigten Werte den Standardwerten entsprechen. Der Emulator NetEm besitzt nicht die

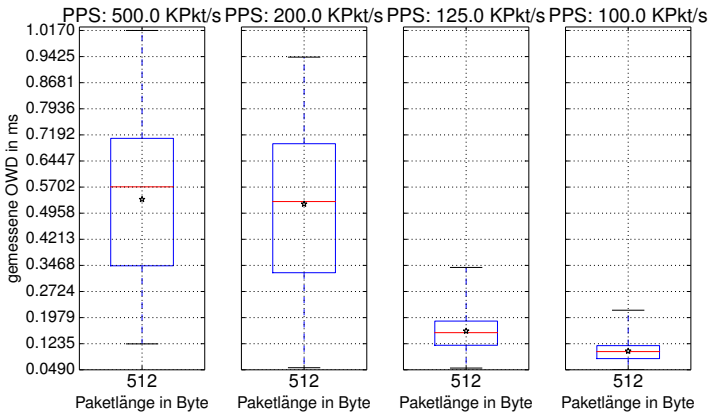


Abbildung 8.2: Gemessene Verzögerungswerte zur Ermittlung der MLFFR am Beispiel von FreeBSD 7.3 (KauNet).

		Paketgröße in Byte	
		64	**1024**
Paketabstand	**20**	48.98	49.06
in μs	**120**	49.37	47.95

Tabelle 8.5: Berechnete Puffergröße des Emulators KauNet für die jeweilige Paketstromeigenschaft.

Möglichkeit Datenraten zu beschränken, vielmehr werden dafür andere *qdiscs* (vgl. Abschnitt 3.2.5) wie z. B. HTB (*Hierarchical Token Bucket*) oder TBF (*Token Bucket Filter*) genutzt. Bei der Ratenbeschränkung durch diese beiden Methoden wird u. a. die Puffergröße als Eingabeparameter verlangt.

8.2 Messungen

Die Ergebnisse werden in Abschnitte aufgeteilt, welche sich jeweils auf eine bestimmte Metrik beziehen. Je nach unterstützter Metrik werden dabei zunächst einzelne ausgewählte Ergebnisse der Emulatoren beispielhaft präsentiert und erläutert. Anschließend werden diejenigen Emulatoren für ausgewählte Strom- und Pfadeigenschaften in einen Zusammenhang gebracht, welche diese Eigenschaften unterstützen.

	Puffergröße / Anzahl Pakete
Linktropy	∼250
Dummynet	∼50
KauNet	∼50

Tabelle 8.6: Puffergröße des jeweiligen Emulators.

8.2.1 Datenrate

Der Hardware-Emulator setzt die Datenratenbeschränkungen von allen getesteten Emulatoren mit den geringsten Abweichungen um. Das liegt zum einen am größeren Puffer und zum anderen an dessen höherer Verarbeitungsgeschwindigkeit, welche wiederum aus einer höheren Taktrate resultiert.

In Abbildung 8.3a werden als Beispiel die relativen Abweichungen der gemessenen Datenraten von den eingestellten Vorgaben am Emulator als Balkendiagramme dargestellt. Die Rate der Probeströme entsprechen dabei der festgelegten MLFFR, d. h. es werden Pakete mit einer Größe von 512 Byte und einer Paketrate von 125 KPkt/s zur Messung verwendet. Zum Vergleich befinden sich auch die relativen Abweichungen der anderen beiden Emulatoren in der Abbildung. DummyNet und KauNet beschränken die Datenraten mit deutlich höheren Abweichungen, wobei gilt, dass die absoluten relativen Abweichungen mit zunehmendem Unterschied zwischen der anliegenden Proberate und der zu emulierenden Datenrate annähernd gleich bleiben.

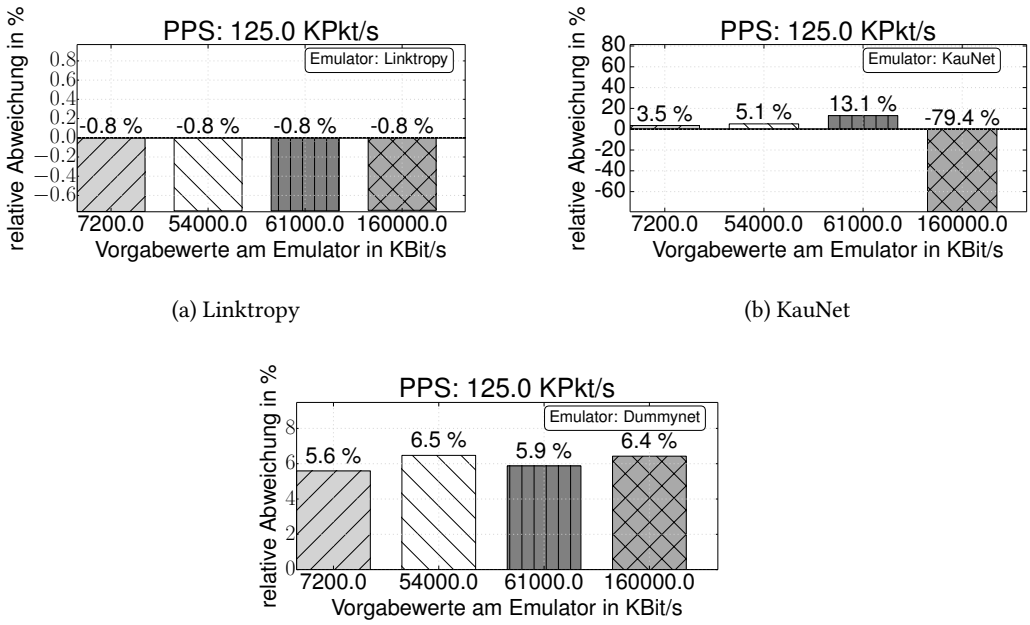
Eine Ausnahme bildet jedoch KauNet. So ist eine besonders starke Abweichung für eine Beschränkung von 160 MBit/s zu erkennen. Der Grund liegt für diesen Fall am fehlerhaften Mustergenerator bzw. am Datentyp der Variable, welche den Wert der Beschränkung aufnimmt. Bei der Eingabe von Datenraten größer als 65535 KBit/s findet ein Überlauf statt und die Einschränkungen beginnen wieder bei 1 KBit/s. Somit liegt der obere Wertebereich für KauNet bereits bei 65535 KBit/s.

Gleichzeitig werden die Auswirkungen einer geringeren Puffergröße deutlich. Für die einzelnen Vorgabewerte reicht die Vorhaltekapazität bei beiden Software-Emulatoren nicht aus und es werden Pakete verworfen. Der Hardware-Emulator dagegen speichert alle ankommenden Pakete und vergrößert deren zeitlichen Abstand. Offenbar gibt es für die Ratenbeschränkung zwei Ansätze. Entweder wird die Rate durch Paketverluste oder aber durch Verzögerungen erwirkt.

Die Genauigkeit der Umsetzung zur Ratenbeschränkung sinkt mit steigender Proberate und damit Last am Emulator. In Abbildung 8.4 werden die Ergebnisse für sinkende PPS und verschiedene Paketgrößen dargestellt. Bereits ab einer PPS von 50 KPkt/s fallen die Abweichungen bis auf wenige Prozent. Lediglich bei einer Paketgröße von 64 Byte sind noch starke Abweichungen zu erkennen. Der Grund liegt in der nötigen Verarbeitungsgeschwindigkeit pro Paket. Viele kleine Pakete bedeuten mehr Verwaltungs- bzw. Behandlungsroutinen als wenige große. Dennoch treten auch für diesen Raten bei den Software-Emulatoren ungewollte Paketverluste auf. Für den möglichen Wertebereich bedeutet dies, dass neben einem Toleranzwert für die Abweichungen auch eine Toleranz für ungewollte Paketeffekte berücksichtigt werden muss.

Für die Betrachtung der Granularität gilt dieses ebenfalls. Die Emulatoren beschränken die anliegende Datenrate annähernd auf die geforderte Rate, jedoch treten ungewollte Paketeffekte auf. In Anhang A.5 sind die Abweichungen für die minimale gewählte Datenrate mit einem Pfad der Länge eins dargestellt. Die Abweichungen bleiben auch bei einer Pfadlänge von 10 näherungsweise gleich (vgl. Anhang A.5). Der Pfad besteht dabei aus neun Abschnitten ohne jedwede Beschränkung und einem einzelnen mit Ratenbeschränkung.

Fazit Somit lässt sich zusammenfassen, dass die Größe des Puffers über die Anzahl an ungewollten Paketeffekten und die Höhe der Differenz aus anliegender und eingestellter Datenrate über die Höhe der Abweichung entscheidet. Aussagen zur Granularität oder zum Wertebereich sind nur unter der Berücksichtigung von vordefinierten Toleranzen möglich. Lediglich der



(a) Linktropy

(b) KauNet

(c) Dummynet

Abbildung 8.3: Auszug aus den Messergebnissen zur Betrachtung der Genauigkeit bei der Umsetzung von verschiedenen vorgegebenen Datenraten für einen einzelnen Pfadabschnitt. Die Paketgröße beträgt bei allen Darstellungen 512 Byte.

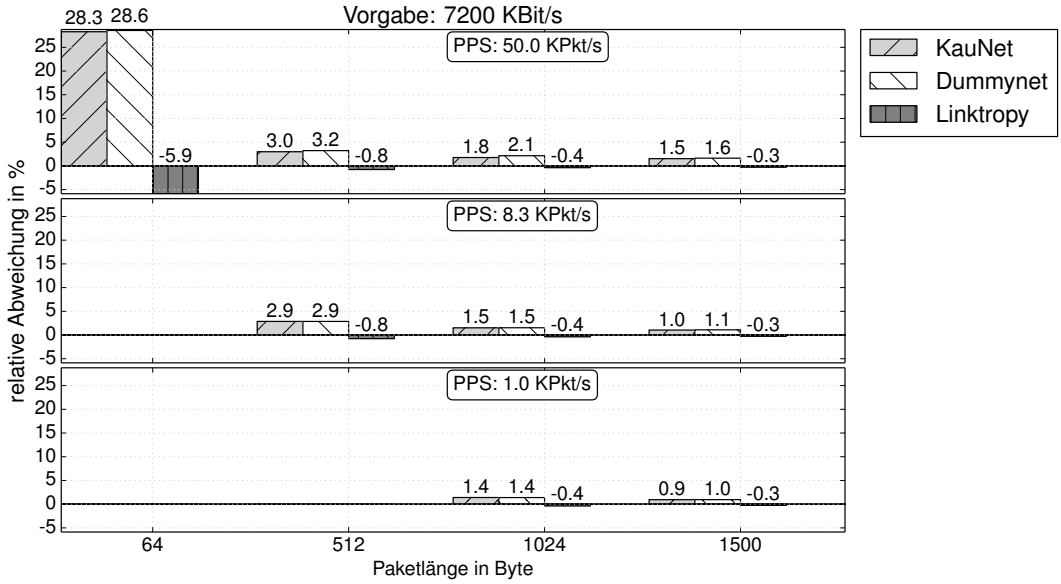


Abbildung 8.4: Relative Abweichungen der gemessenen Datenraten bei unterschiedlichen Stromeigenschaften und einem einzelnen Pfadabschnitt. Die vermeidlich fehlenden Balken in der Abbildung kommen dadurch zustande, dass mit der jeweiligen Kombination von Paketgröße sowie -abstand nicht die geforderte Datenrate erreicht wird. Vorgabe: 7200 KBit/s

Hardware-Emulator liegt für Paketgrößen oberhalb von 512 Byte innerhalb des vorher festgelegten Grenzbereichs.

8.2.2 Verzögerung

Bei den Messungen zur Genauigkeit von emulierten Verzögerungen heben sich besonders der Linktropy und NetEm durch ihre Stabilität der Werte hervor. Als Beispiel werden die CoDVwB-MA-Werte für die kritische Paketgröße 64 Byte mit einer Rate von 50 KPkt/s in Abbildung 8.5 betrachtet. Die Variationen der Verzögerungen fallen beim Hardware-Emulator und bei NetEm deutlich geringer aus als bei Dummynet und KauNet. Noch über den Betrachtungszeitraum von einer Millisekunde hinaus variieren die Verzögerungen mit einem Betrag von fünf Prozent um den Erwartungswert herum. Erst bei einem Betrachtungszeitraum von $\sim$ 100 ms bleiben die Werte nach außen hin stabil.

Diese Schwankungen werden ebenfalls durch die Betrachtung der CDF in Anhang A.6 sichtbar. Zusätzlich befinden sich in diesem Anhang die Verteilungsfunktionen der anderen Stromeigenschaften. Auch bei diesen sind diese Schwankungen zu erkennen. Der Grund dafür ist beim Taktgeber zu finden. Bei KauNet und Dummynet bzw. deren Betriebssystemen ist standardmäßig eine Taktrate von 1000 Hz eingestellt. Dadurch liegen die minimalen Paketverzögerungen von beiden Emulatoren bei etwas über 1 ms pro Paket. Bei höheren Vorgabewerten fällt dieser Punkt nicht mehr so stark ins Gewicht.

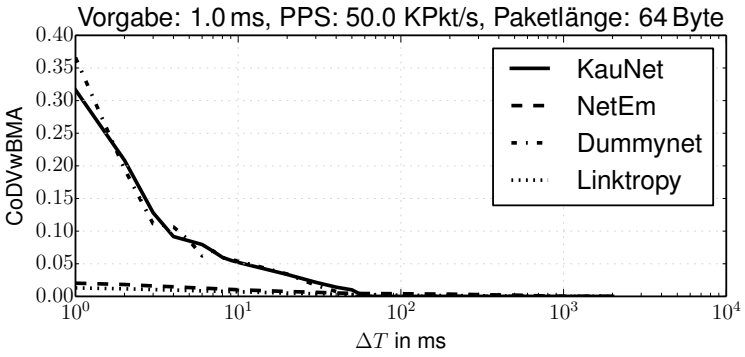


Abbildung 8.5: Auszug aus den berechneten CoDVwBMA-Werten zur Ermittlung der Wertstabilität des jeweiligen Emulators während der Messung der Verzögerungen. Vorgabe: 1 ms

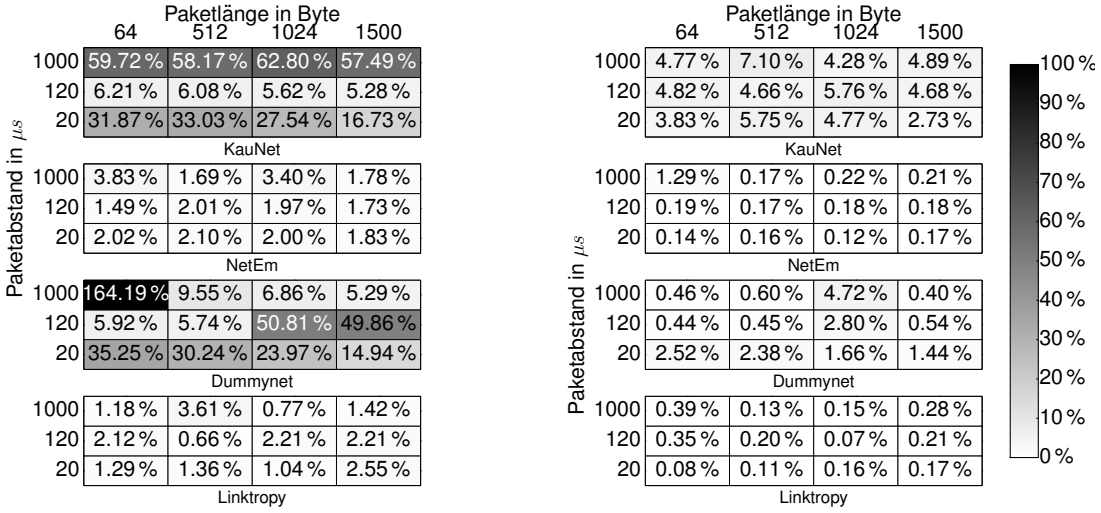
In Anhang A.6 ist gleichzeitig die Genauigkeit ablesbar. So zeigt sich, wie bei den Untersuchungen zur Grundverzögerung, dass die Genauigkeit abhängig von der Paketgröße ist. Je größer ein Paket ist, desto deutlicher weichen die Werte von der Vorgabe ab. Konkrete Wertangaben bzgl. dem CoV befinden sich in den Tabellen 8.7, welche als *Heatmap* dargestellt sind. Je dunkler eine Zelle in der Tabelle ist, desto größer ist der Wert des Variationskoeffizienten und damit die Abweichung vom Erwartungswert bzw. der Vorgabe.

Für einen Vergleich werden in der selben Übersicht die CoV-Werte der Verzögerungen für die Vorgaben 10 ms, 100 ms und 1000 ms gezeigt. Hierbei sind die Abweichungen für die Software-Emulatoren deutlich zurückgegangen, was zum einen auf die geringere Gewichtung der Werte in Bezug auf den hohen Erwartungswert und zum anderen auf die steigende Unabhängigkeit von der Taktrate zurückzuführen ist.

In Bezug auf die Pfadverkettungen ist auch hierbei eine Abhängigkeit von der Puffergröße zu erkennen. Liegt der Paketstand um ein Vielfaches niedriger als die Verzögerung so steigt die Gefahr, dass die Puffer nicht mehr ausreichen, da zunächst alle auftreffenden Pakete in den *Pipes* zwischengespeichert werden müssen. Speziell bei KauNet treten hohe Paketverluste auf, wodurch eine Betrachtung und ein Vergleich der Messwerte hinfällig wird.

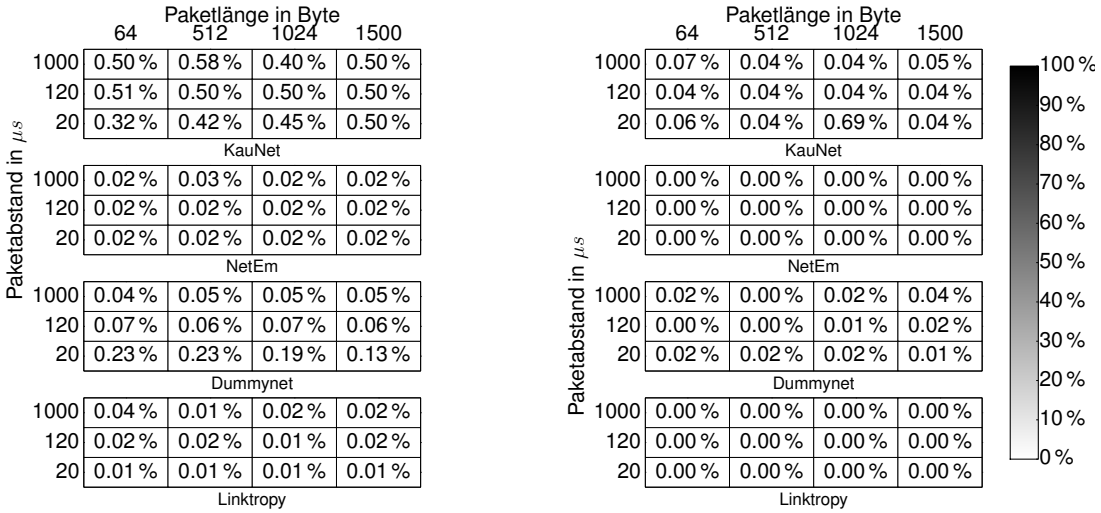
Anders verhält es sich dagegen bei Dummynet. Die Werte für die Untersuchungen mit zehn Pfadabschnitten, welche jeweils eine Verzögerung von 10 ms verursachen, sind in Abbildung 8.6 zu sehen. Alle Werte liegen annähern bei 100 ms, was dem erwarteten Resultat bei zehn hintereinander geschalteten Abschnitten entspricht.

Fazit Somit lässt sich zusammenfassen, dass der Taktgeber ausschlaggebend für die Genauigkeit, Granularität und den Wertebereich ist. Wobei gleichzeitig die Puffergröße ausreichend groß gewählt werden muss, um bei mehreren Pfadabschnitten genügend Pakete zwischenspeichern zu können. Die Taktgeber des Hardware-Emulators und von NetEm arbeiten ausreichend schnell, um auch Verzögerungen unterhalb 1 ms zu erlauben bzw. um wesentlich stabilere Verzögerungswerte zu liefern. Bei höheren Vorgabewerten liefern alle Emulatoren Verzögerungen innerhalb des Toleranzbereichs von $\pm 10\,\%$ relative Abweichung. Als einziger Emulator erlaubt Dummynet



(a) Vorgabe 1 ms

(b) Vorgabe 10 ms



(c) Vorgabe 100 ms

(d) Vorgabe 1000 ms

Tabelle 8.7: Abweichungen der mittleren emulierten Verzögerung bei verschiedenen Stromeigenschaften des jeweiligen Emulators von der Vorgabe. Pfadlänge: 1

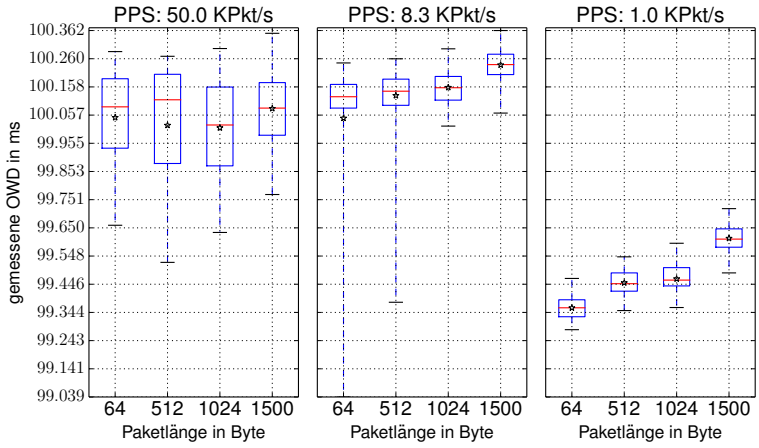
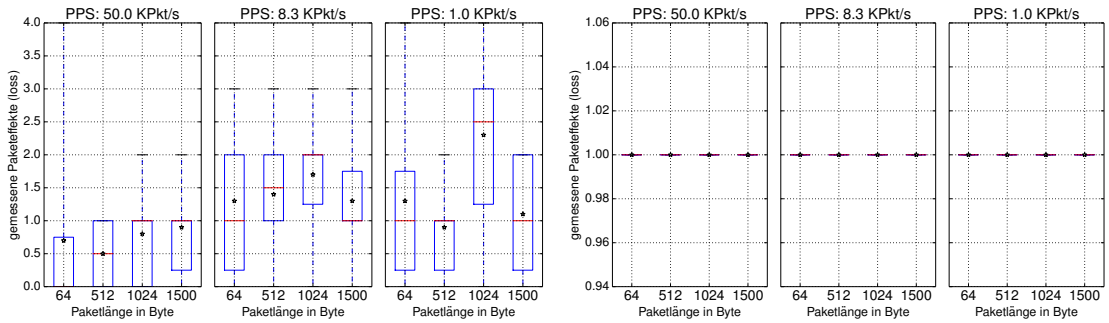


Abbildung 8.6: Gemessene Verzögerungen mit dem Emulator DUMMYNET als Box-Diagramm für die verschiedenen Paketlängen und -raten bei 10 Pfadabschnitten. Vorgabe: 10 ms

dabei sogar Pfadverkettungen. Der untersuchte Maximalwert von 1000 ms wurde ebenfalls bei allen Emulatoren mit positiven Ergebnis gemessen.

8.2.3 Paketeffekte

Verlust KAUNET emuliert aufgrund seiner deterministischen Paketauswahl die stabilsten Werte und die exakte Anzahl an Paketverlusten bzw. -neuordnungen. Zur Gegenüberstellung sind in Abbildung 8.7 die Messergebnisse des Hardware-Emulators und von KAUNET bei einer Vorgabe von 0,001 % Verlust zu sehen. Während beim Hardware-Emulator noch deutliche Abweichungen



(a) Linktropy (b) KauNet

Abbildung 8.7: Auszug aus den Messergebnissen zur Betrachtung der Genauigkeit bei der Umsetzung von 0,001 % Paketverlust für einen einzelnen Pfadabschnitt. Die Box-Diagramme von KAUNET erscheinen wegen der fehlenden Abweichung als einzelne Striche mit einem Stern (Durchschnittswert) in der Mitte.

in der Verlustanzahl zwischen den einzelnen Probedurchläufen zu sehen sind, entsprechen

die Messwerte bei KauNet exakt der Vorgabe. Bei keiner der verwendeten Paketlängen oder -größen treten bei KauNet Abweichungen auf. In Abbildung 8.8 sind beispielsweise die erfassten Paketverluste für die unterschiedlichen verwendeten Stromeigenschaften bei einer Vorgabe von 1 % dargestellt. Zudem werden in dieser Abbildung die Ergebnisse der anderen Emulatoren

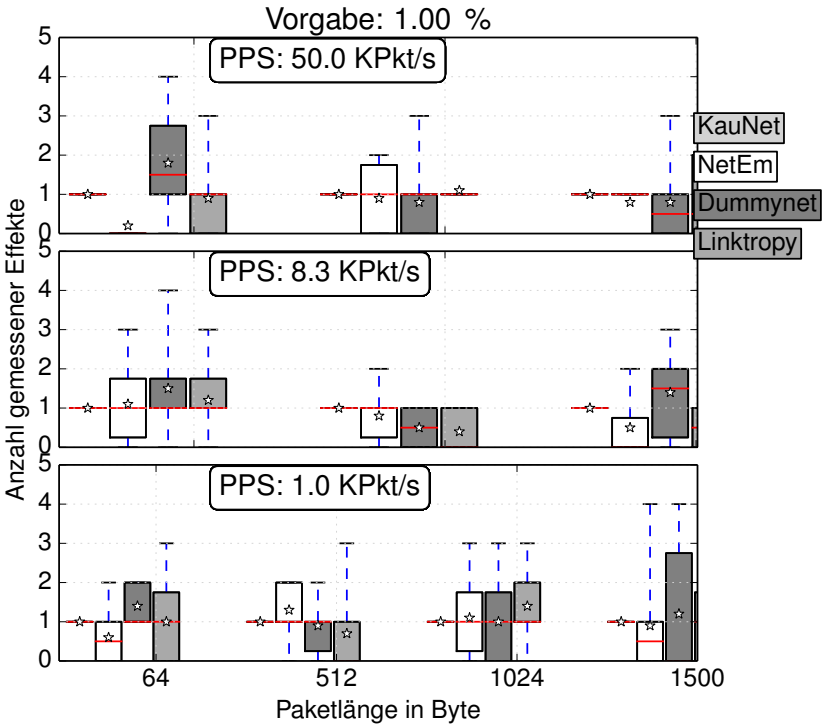


Abbildung 8.8: Erfasste Paketverluste als Box-Diagramme für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt. Auch hier erscheinen die Box-Diagramme von KauNet wegen der fehlenden Abweichung als einzelne Striche mit einem Stern in der Mitte.

gezeigt. Bei diesen treten die Verluste unregelmäßiger und mit Abweichungen zum Vorgabewert auf.

In Bezug auf die Granularität bleiben dennoch alle Emulatoren innerhalb der Toleranzgrenzen von ± 3 % relative Abweichung. Es sei jedoch angemerkt, dass mehrere Probedurchläufe erforderlich sind, um insgesamt das benötigte Verhältnis zwischen Verlusten und keinen Paketverlusten zu erhalten. Die Ursache ist der jeweilige Pseudozufallsgenerator, welcher beispielsweise die ersten zehn Pakete verwirft und erst nach 10000 Paketen einen weiteren Verlust entscheidet. Werden dabei ein zu kleines Beobachtungsintervall und zu wenige Probedurchläufe gewählt, entstehen starke Abweichungen vom erwarteten Wert.

Dummynet ist auch hierbei der einzige Emulator, welcher zuverlässig Pfadverkettungen erlaubt. In Abbildung 8.9 ist das Messergebnis dargestellt. Die Werte befinden sich innerhalb des Erwartungsbereiches. D. h., bei zehn Abschnitten mit der gleichen Verlustrate von 1 %, muss die resultierende Rate bei 9,6 % liegen.

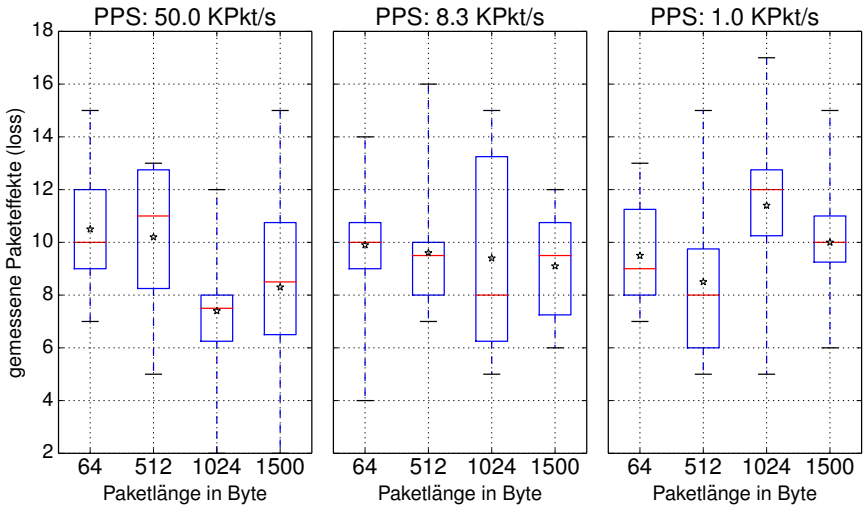


Abbildung 8.9: Erfasste Paketverluste als Box-Diagramme für die verschiedenen Paketlängen sowie -raten bei 10 Pfadabschnitten und mit Dummynet als Emulator. Der Erwartungswert liegt bei 9,6 Verlusten.

Ein anderer Emulator, der dieses ebenfalls durchführen sollte, ist KauNet. Jedoch treten bei diesem schwerwiegende Ausnahmefehler im Kernel von FreeBSD auf und verursachen Systemabstürze, sobald mehrere Pfadabschnitte konfiguriert werden. Daher werden auf weitere Untersuchungen mit KauNet für mehrere Pfadabschnitte verzichtet.

Neuordnung Auch hierbei hebt sich KauNet von den anderen Emulatoren ab. Ein direkter Vergleich kann beispielsweise mit Hilfe von Abbildung 8.10 durchgeführt werden. Ein weiteres Beispiel befindet sich ebenfalls in Anhang A.7. KauNet setzt exakt die Vorgabe um. Wogegen die Werte von z. B. NetEm deutliche Abweichungen aufgrund der Zufallsentscheidung aufweisen. Für eine PPS von 50 KPkt/s liegen die Werte deutlich oberhalb der Vorgabe und für eine Rate von 1 KPkt/s finden gar keine Neuordnungen statt. Daher ist zu vermuten, dass die Zufallsentscheidungen nicht pro Paket sondern pro Zeiteinheit getroffen werden. Was wiederum darauf hindeutet, dass das Betrachtungsintervall zu klein und zu wenige Probedurchläufe gewählt wurde. Einzig der Hardware-Emulator weist ähnliche Werte in Bezug auf die mittlere Neuordnungsrate auf. Die Entscheidungszeitpunkte scheinen bei diesem Emulator mit einem geringeren Abstand zu erfolgen.

Dopplung NetEm besitzt als Einziger, gegenüber den anderen Software-Emulatoren, die Fähigkeit Paketdopplungen zu emulieren. Jedoch weichen auch bei der Paketdopplung die gemessenen Paketeffekte von den vorgegebenen ab. So Schwanken beispielsweise bei einer Vorgabe von 0,01 % die Anzahlen der Paketeffekte zwischen 0 % und 0,08 % (vgl. Abb. 8.11). Die Mittelwerte liegen dabei bei 0,02 % und somit innerhalb des vorher definierten Toleranzbereichs.

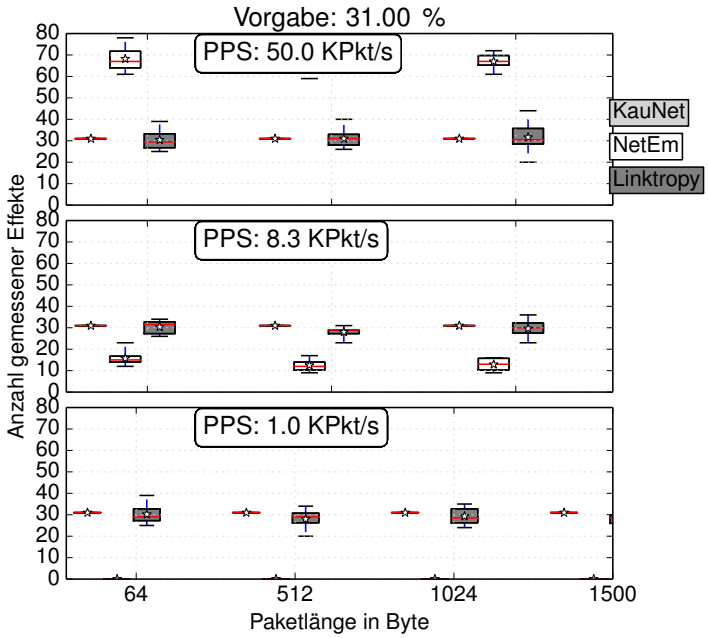


Abbildung 8.10: Erfasste Paketneuordnungen als Box-Diagramme für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt.

Fazit Aus den Ergebnissen lässt sich zusammenfassen, dass deterministische Vorgaben den Vorteil haben, nicht von bestimmten Betrachtungszeiträumen und Probeanzahlen abhängig zu sein. So stimmen die emulierten Paketverluste und -neuordnungen bei KauNet zu 100 % mit den Vorgaben überein. Für geringe Vorgabewerte liegen auch die anderen Emulatoren bei allen drei Paketeffekten innerhalb des Toleranzbereichs. Dies ändert sich jedoch für steigende Vorgaben bei der Paketneuordnung, da die Zufallsgeneratoren diesen Effekte zeitgetrieben steuern. Ein zu hohe Rate sorgt für zu viele Neuordnungen, wohingegen eine zu niedrige für zu wenige Paketeffekte sorgt. Abhängigkeiten von der Paketgröße konnten nicht festgestellt werden.

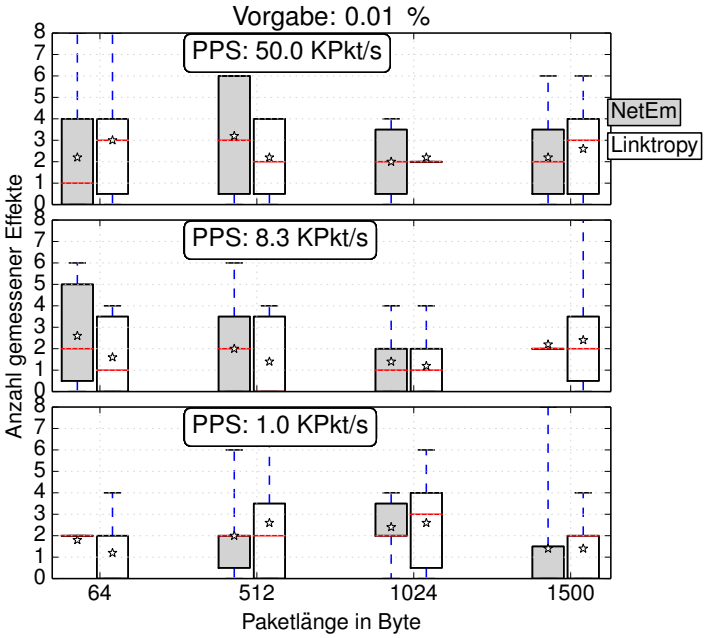


Abbildung 8.11: Erfasste Paketdopplungen als Box-Diagramme für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt.

9 Zusammenfassung und Ausblick

Das Thema dieser Arbeit ist der Entwurf und die Umsetzung eines Messkonzeptes zur Analyse von verschiedenen Netzwerkemulatoren in Bezug auf ihre Genauigkeit und Leistungsfähigkeit bei der Nachbildung von bestimmten Paketeffekten, der Einhaltung von Verzögerungen sowie der Stabilität bei der Beschränkung von Datenraten. Das Ergebnis sind Daten aus denen sich Informationen für Vergleichszwecke von verschiedenen Netzwerkemulatoren ableiten lassen.

Um Einflussfaktoren auf die genannten Eigenschaften zu ermitteln, werden zunächst die wesentlichen Prinzipien von Netzwerkemulatoren sowie die Netzwerkeinflüsse auf Datenpakete im Grundlagenkapitel 2 vorgestellt. Gleichzeitig wird in diesem Kapitel der Begriff der Emulation definiert und von der *Simulation* sowie *SEmulation* abgegrenzt. Aus diesen Grundlagen ergibt sich ein Klassifikationsmodell für Emulatoren, bestehend aus feststehenden und messbaren Eigenschaften. Dieses Modell kann dazu genutzt werden, anhand von Randbedingungen an eine mögliche Testumgebung, den passenden Emulator auszuwählen. Gleichzeitig lassen sich so ebenfalls Gemeinsamkeiten, wie z. B. die Unterstützung eines bestimmten Paketeffektes finden. Diese Übereinstimmungen bieten die ersten Ansatzpunkte für ein einheitliches Messkonzept, mit dem sich mehrere Emulatoren untereinander vergleichen lassen.

Weitere Ansatzpunkte ergeben sich durch die Auswertung von vergleichbaren Arbeiten. Die Literaturrecherche ergab jedoch, dass nur wenige ähnliche Arbeiten mit Vergleichsstudien existieren. Vielmehr bieten die Entwickler des jeweiligen Emulators in ihren Dokumentationen Vorgehensweisen zur Analyse von Emulatoren an. Allerdings beschränken sich diese auf bestimmte Merkmale ihres Emulator. Im Kapitel 3 zur Analyse von verwandten Arbeiten werden daher neben den Vergleichsstudien auch Dokumentationen betrachtet. Die Wahl der jeweiligen Dokumentation stützt sich dabei u. a. auf eine Trendanalyse von Internet-Suchanfragen bzgl. dem Stichwort Netzwerkemulator und der Anzahl der Zitation. Die Analysen ergeben, dass im Blickfeld der Arbeiten hauptsächlich die Emulatoren DummyNet, NetEm, KauNet, WanEm, NetPath, Immunes und ModelNet liegen.

Insgesamt werden zwölf Arbeiten in Kapitel 3 vorgestellt, deren Schwerpunkte auf einem oder mehreren der genannten Emulatoren liegen. Aus diesen lassen sich viele Ansätze zur Messung von bestimmten Eigenschaften von Emulatoren für das eigene Messkonzept gewinnen. Dazu zählen beispielsweise ein Versuchsaufbaus mit passiver Paketerfassung und Metriken wie dem CoTV.

Zusammen mit dem Klassifikationsmodell aus Kapitel 2 und den gewonnenen Erkenntnissen aus Kapitel 3 ergibt sich eine grobe Struktur des eigenen Konzeptes, für das im Kapitel 4 eine Anforderungsanalyse durchgeführt wird. Als Ergebnis dieser Analyse entstehen mehrere Teilziele, gruppiert nach funktionalen und qualitativen Anforderungen, welche sich wiederum in weitere Teilschritte aufgliedern. Zu diesen Teilzielen gehört u. a. ein Hardware- sowie Softwaretechnischer Versuchsaufbau, der es ermöglicht Pakete zu generieren, *uhrensynchron* zu erfassen und bzgl. bestimmter Metriken zu untersuchen.

Im nachfolgenden Kapitel 5 werden entsprechend der gefassten Teilziele die einzelnen Konzepte zu deren Umsetzung vorgestellt. Für den Hardware-technischen Versuchsaufbau werden die Testumgebungen aus den vorgestellten Vergleichsstudie sowie Dokumentationen betrachtet und gegenüber gestellt. Als Resultat entsteht eine eigene Lösung, welche den Sender und Empfänger der Probepakete auf einem einzelnen Messrechner vereint. Dadurch entfällt die benötige Uhrensynchronisation und es können exakte Verzögerungswerte gemessen werden. Ein weiterer Rechner dient als Versuchsrechner, auf welchem die Emulatoren installiert werden. Mögliche Engstellen werden mit verschieden Mitteln vermieden. Der Software-technische Aufbau besteht aus mehrere Modulen die zusammen die jeweilige Untersuchung automatisiert starten, steuern, durchführen, beenden und auswerten. Eine Versuchssteuerung nimmt die nötigen Konfigurationen bzgl. der Netzwerkeinstellungen und Versuchsparameter entgegen und startet zunächst jeweils eine Paketerfassung für den Sender sowie Empfänger. Anschließend wird für jeden Probedurchlauf ein Paketgenerator mit bestimmten Paketstromeigenschaften konfiguriert und ebenfalls gestartet. Die Messwerte werden während der Untersuchungen in Dateien gespeichert und nach Beendigung eines Versuchsdurchlaufes ausgewertet sowie visualisiert. Dabei kann auf Lösungen aus einer vorangegangenen Arbeit zurückgegriffen werden. Diese Lösungen umfassen die Paketgenerierung und -erfassungen sowie einige Analysemethoden zur Wertermittlung von Netzwerkmetriken. Wobei jedoch im Einzelnen Verbesserungen und Anpassungen vorgenommen werden.

Weiterhin werden in diesem Kapitel die Vorgehensweisen bei den einzelnen Messungen zunächst allgemein und anschließend detailliert beschrieben. Die Genauigkeit eines Emulators misst sich anhand der Abweichung des gemessenen vom vorgegebenen Wert. Die Leistungsfähigkeit wird durch aufsteigende Paketstrom- und Pfadeigenschaften, wie z. B. der Paketrate oder Pfadlänge, ermittelt. Gleichzeitig wird eine neue Metrik zur Betrachtung der Wertstabilität entwickelt. Durch diese ist es möglich Aussagen über Abweichungen für verschiedene Zeitspannen zu treffen und damit denjenigen Betrachtungszeitraum zu erhalten, bei dem die geringsten Schwankungen auftreten. Dabei gilt je kleiner dieser Zeitraum ist, desto stabiler verhalten sich die Messwerte.

Das sechste Kapitel beinhaltet ausgewählte Teile der Umsetzung und Implementation des Messkonzepts. In diesem Zusammenhang werden der reale Versuchsaufbau und einige Programmdetails der Analyse-Software vorgestellt. Außerdem werden in diesem Kapitel die konkreten Wertangaben der Versuchsparameter erläutert. Zu diesen Parametern gehören beispielsweise die verwendeten Paketstromeigenschaften und die Vorgabewerte am Emulator.

Der Nachweis über die korrekte Funktion der Implementationen wird im siebenten Kapitel erbracht. Dazu werden zunächst der Versuchsaufbau und anschließend der Paketgenerator sowie die Datenerfassung mit Hilfe der Messwerkzeuge Wireshark und ping getestet. Dabei zeigt sich die hohe Präzision mit der die Generierung und Erfassung arbeiten.

Im letzten Kapitel werden abschließend mit dem entwickelten Messkonzept die vier Emulatoren Dummynet, KauNet, Netem und Linktropy bezüglich ihrer Genauigkeit und Leistungsfähigkeit bei der Nachbildung von Verzögerungen, Datenraten, Paketverlusten, -dopplungen sowie -neuordnungen untersucht.

Der komplette Umfang der Messergebnisse kann dabei nicht dargestellt werden, da deren Anzahl außerhalb des Rahmens dieser Arbeit liegt. Vielmehr dienen die Durchführungen im letzten Kapitel als Beispiel für die Leistungsfähigkeit des entwickelten Messkonzepts. Allein für diese vier

Emulatoren wurden über 92000 Einzelmessungen automatisiert durchgeführt und anschließend ausgewertet. Das Messkonzept erlaubt es, sich einen Überblick über die messbaren Eigenschaften eines Netzwerkemulators zu verschaffen und unterstützt dabei eine Vielzahl von Netzwerkmetriken. Zudem ist es in der Lage, die Messergebnisse eines Emulators mit denen von anderen in einen Zusammenhang zu bringen und mit verschiedenen Diagrammen grafisch darzustellen. Da der Emulator dabei als *Black-Box* angesehen wird, kann die Testumgebung darüber hinaus zur Messung von realen Verbindungen verwendet werden.

Neben den bereits durchgeführten Untersuchungen können zukünftig noch weitere Netzwerkemulatoren analysiert werden, um so zusammen mit dem entwickelten Klassifikationsschema einen einheitlichen und umfassenden Überblick über deren Eigenschaften zu gewinnen.

In Bezug auf die Weiterentwicklung des vorgestellten Messkonzepts können durch dessen modularen Aufbau künftig weitere Metriken, z. B. aus dem Bereich des Mobilfunks, hinzugefügt werden, um das Spektrum der Emulatoren zu erweitern. Außerdem gestattet dieser Aufbau einen Austausch von Komponenten, wie beispielsweise dem Paketgenerator. Dadurch ist eine stetige Verbesserung der Kernkomponenten möglich. Eine weitere Verbesserungsmöglichkeit bieten grafische Oberflächen zur Konfiguration der Versuchsparameter. Aktuell werden diese in Form von Textdateien an das System übergeben. In der derzeitigen Version ist das Messsystem noch von bestimmten spezifischen Betriebssystemfunktionen abhängig, daher bietet es sich an entweder betriebssystemunabhängige Funktionen zu nutzen oder mehrere Versionen für unterschiedliche Plattformen zu erstellen.

Es bestehen viele weitere Möglichkeiten an das Messkonzept anzuknüpfen. Das Ergebnis dieser Arbeit bietet alle notwendigen Konzepte und Implementationen für die Genauigkeits- und Leistungsanalyse von Netzwerkemulatoren. (und das soll mir erstmal jemand nachmachen :-))

Anhang

A.1 Nachweis Versuchsaufbau: Ping-antworten 120
A.2 Konfigurationsdatei für die Netzwerkeinstellungen 121
A.3 Konfigurationsdatei für die Angabe von Messparametern 122
A.4 Vorfelduntersuchungen: Grundverzögerungen 123
A.5 Messung: Datenrate 124
A.6 Messung: Verzögerungen 125
A.7 Messung: Neuordnung 126
A.8 Erklärung zum Urheberrecht 127

A.1 Nachweis Versuchsaufbau: Ping-antworten

```
C:\>ping -S 10.0.0.2 -n 5 -l 100 -i 2 10.0.0.1

Pinging 10.0.0.1 from 10.0.0.2 with 100 bytes of data:
Reply from 10.0.0.1: bytes=100 time<1ms TTL=64
Reply from 10.0.0.1: bytes=100 time<1ms TTL=64
Reply from 10.0.0.1: bytes=100 time<1ms TTL=64
Reply from 10.0.0.1: bytes=100 time<1ms TTL=64
Reply from 10.0.0.1: bytes=100 time<1ms TTL=64

Ping statistics for 10.0.0.1:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping -S 10.0.0.2 -n 5 -l 100 -i 2 10.0.0.5

Pinging 10.0.0.5 from 10.0.0.2 with 100 bytes of data:
Reply from 10.0.0.5: bytes=100 time<1ms TTL=64
Reply from 10.0.0.5: bytes=100 time<1ms TTL=64
Reply from 10.0.0.5: bytes=100 time<1ms TTL=64
Reply from 10.0.0.5: bytes=100 time<1ms TTL=64
Reply from 10.0.0.5: bytes=100 time<1ms TTL=64

Ping statistics for 10.0.0.5:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping -S 10.0.0.2 -n 5 -l 100 -i 2 10.0.0.6

Pinging 10.0.0.6 from 10.0.0.2 with 100 bytes of data:
Reply from 10.0.0.6: bytes=100 time<1ms TTL=127
Reply from 10.0.0.6: bytes=100 time<1ms TTL=127
Reply from 10.0.0.6: bytes=100 time<1ms TTL=127
Reply from 10.0.0.6: bytes=100 time<1ms TTL=127
Reply from 10.0.0.6: bytes=100 time<1ms TTL=127

Ping statistics for 10.0.0.6:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

A.2 Konfigurationsdatei für die Netzwerkeinstellungen

```
1  # Configfile -- Network: sender (tx) ->Emulator-> reciever (rx)
2  # ------------------------------------------------------------
3  # Note:
4  # - this file can also be generated by commando
5  # - Tx and Rx have to be placed on the same machine with different subnets.
6  # - Emulator is the gateway
7
8  # S E N D E R
9  tx_device_id = {A97B708B-C2B7-48F2-949A-999829423225}
10 tx_ip_addr = 10.0.0.2
11 tx_mac_addr = 00-0e-0c-c4-e4-e0
12 tx_port = 2606
13
14 # G A T E W A Y (of Sender)
15 gw_ip_addr = 10.0.0.1
16 gw_mac_addr = 00-0e-0c-c3-09-xd          <----- Nachweis 2
17
18 # R E C I E V E R
19 rx_device_id = {A15744E6-3100-4817-8FB6-x} <--- Nachweis 2
20 # rx_ip_addr = 10.0.0.6                   <----- Nachweis 1
21 rx_mac_addr = 68-05-ca-1d-cf-b7
22 rx_port = 1308
```

A.3 Konfigurationsdatei für die Angabe von Messparametern

```
1
2  # Config-file -- Stream (Strm) configuration
3  #-----------------------------------------------------
4  # - Protokoll (Proto) can be UDP
5  # - A Strm consists of a number of Trains (Trn_nr).
6  # - A Trn consists of a number of Packets (Pckt_nr).
7  # - A Packet (Pckt) has a number of bytes (Pckt_size)
8  #   Note: given value cannot contain 'K'-Kilo or 'M'-Mega.
9  # - Each Strm has a sequence-number (Strm_seq_nr).
10 # - The inter-departure time (gap) between Strm, Trains (Trn) and Packets (Pckt) can be given
11 #   explicitly. Whereby Strm_gap are given in 'sec', Trn_gap in 'msec' and Pckt_gap in 'usec'
12 # - Also a minimum duration for Strm (in 'sec') or Trn (also in 'sec') can be configured.
13 #   Note: the number of Trn and Pckt will increase until duration is reached.
14 #   If the duration of Strm_dur is '0', then it will be ignored and the given trn_nr
15 #   or pkt_nr is used.
16 # - A Measurement can be one of the following type (Type):
17 #   - Static (S)              : Strm_seq_nr, Strm_gap, Strm_dur, Trn_nr, Trn_gap, Trn_dur,
18 #                               Pckt_nr, Pckt_gap
19 #                               has to be given explicitly. PPS or bps will ignored
20 #   - Static Extended (E)     : Pckt_gap and Pckt_size can be given as a list
21 #   - Packetrate per Train (P): Strm_gap, Strm_dur, Trn_nr, Trn_gap, Trn_dur, min_PPS,
22 #                               max_PPS, step_PPS has to be given explicitly.
23 #                               Rest will be ignored.
24 #   - Bitrate per Train (B)   : Strm_gap, Strm_dur, Trn_nr, Trn_gap, Trn_dur, min_bps, max_bps,
25 #                               step_bps has to be given explicitly. Rest will be ignored.
26 # Meas., Type, [...], Trn_dur, Pckt_nr, Pckt_gap, Pckt_size      , min_PPS, max_PPS, [...]
27 #    1 ,   S , [...], 1000   , 10 K   , 10      , 64             , -      , -      , [...]
28      2 ,   E , [...], 100    , 10 K   ,         , [64; 512; 1500], -      , -      , [...]
29 #    2 ,   E , [...], 1000   , 10 K   , [20;120], [64; 512; 1500], -      , -      , [...]
30      3 ,   P , [...], 10000  , -      , -       , 1512           , 200 K  , 100 K  , [...]
31 #    5 ,   B , [...], 1000   , -      , -       , 512            , -      , -      , [...]
32
33 # or in a shorter notation
34 # Note:
35 # - columns with '-' will not needed. The type gives the hint.
36 # - if the Measure-number is duplicated the last configuration/row will be used.
37 #       3,S,UDP,3,5,300,10,10,1,100,5,1500
38 #       4,P,UDP,4,1,10,200,1,150,100,150 K,50 k
39 #       5,B,UDP,5,0,10,200,1,1000,1 M, 2 M, 1 M
```

A.4 Vorfelduntersuchungen: Grundverzögerungen

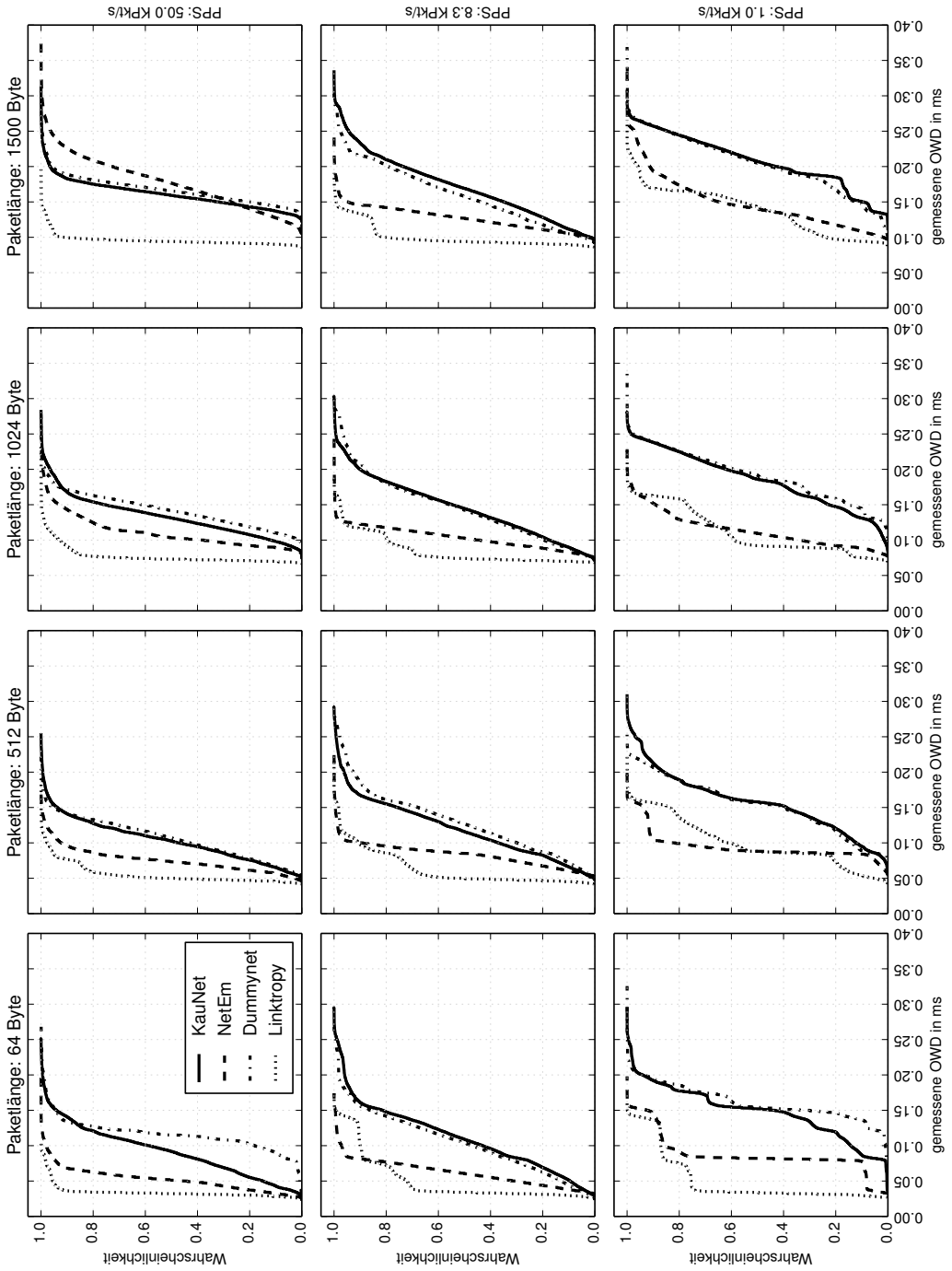


Abbildung A.1: Grundverzögerungen als Verteilungsfunktion des jeweiligen Emulators für die verschiedenen Paketlängen und -raten.

A.5 Messung: Datenrate

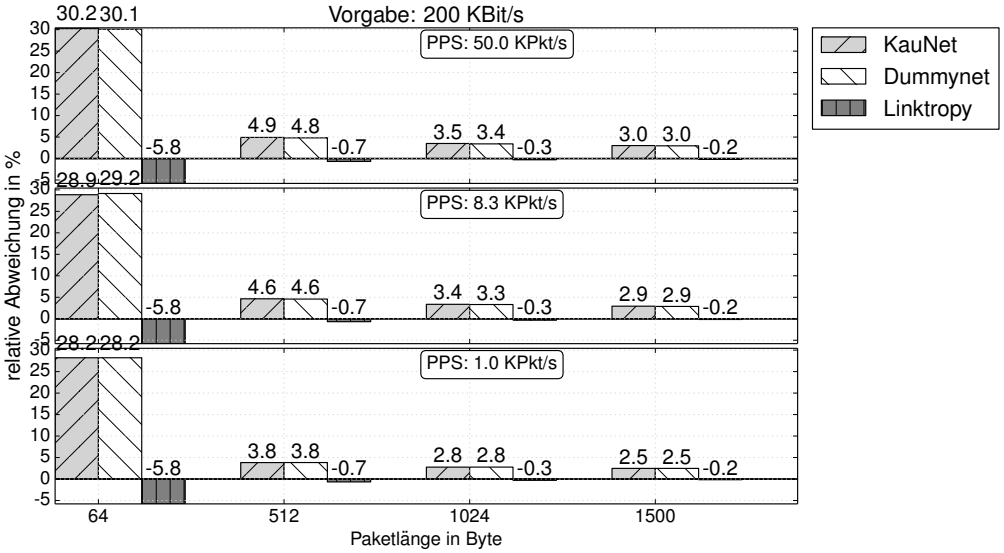


Abbildung A.2: Gemessene Datenraten bei unterschiedlichen Stromeigenschaften und einem einzelnen Pfadabschnitt. Vorgabe: 200 KBit/s

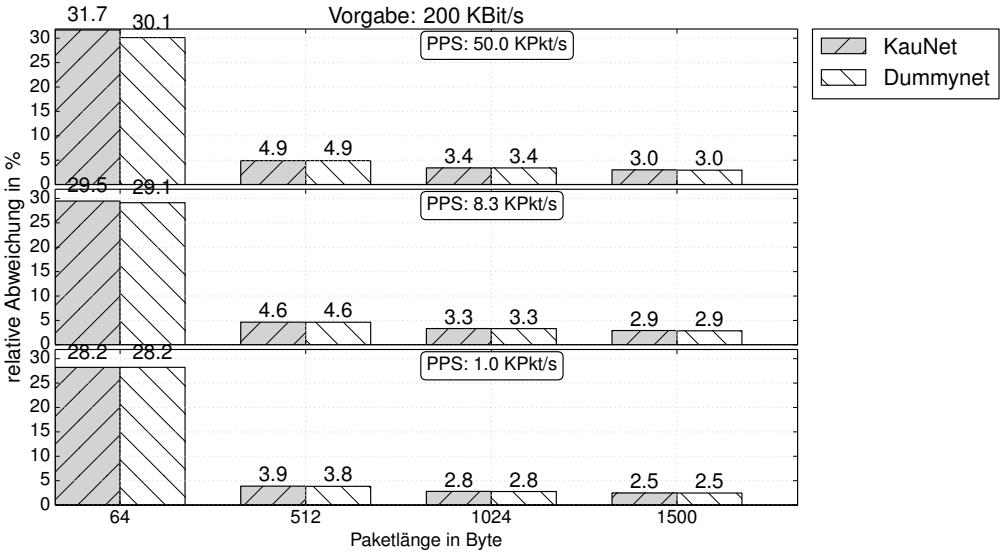


Abbildung A.3: Gemessene Datenraten bei unterschiedlichen Stromeigenschaften und 10 Pfadabschnitten. Vorgabe: 200 KBit/s

A.6 Messung: Verzögerungen

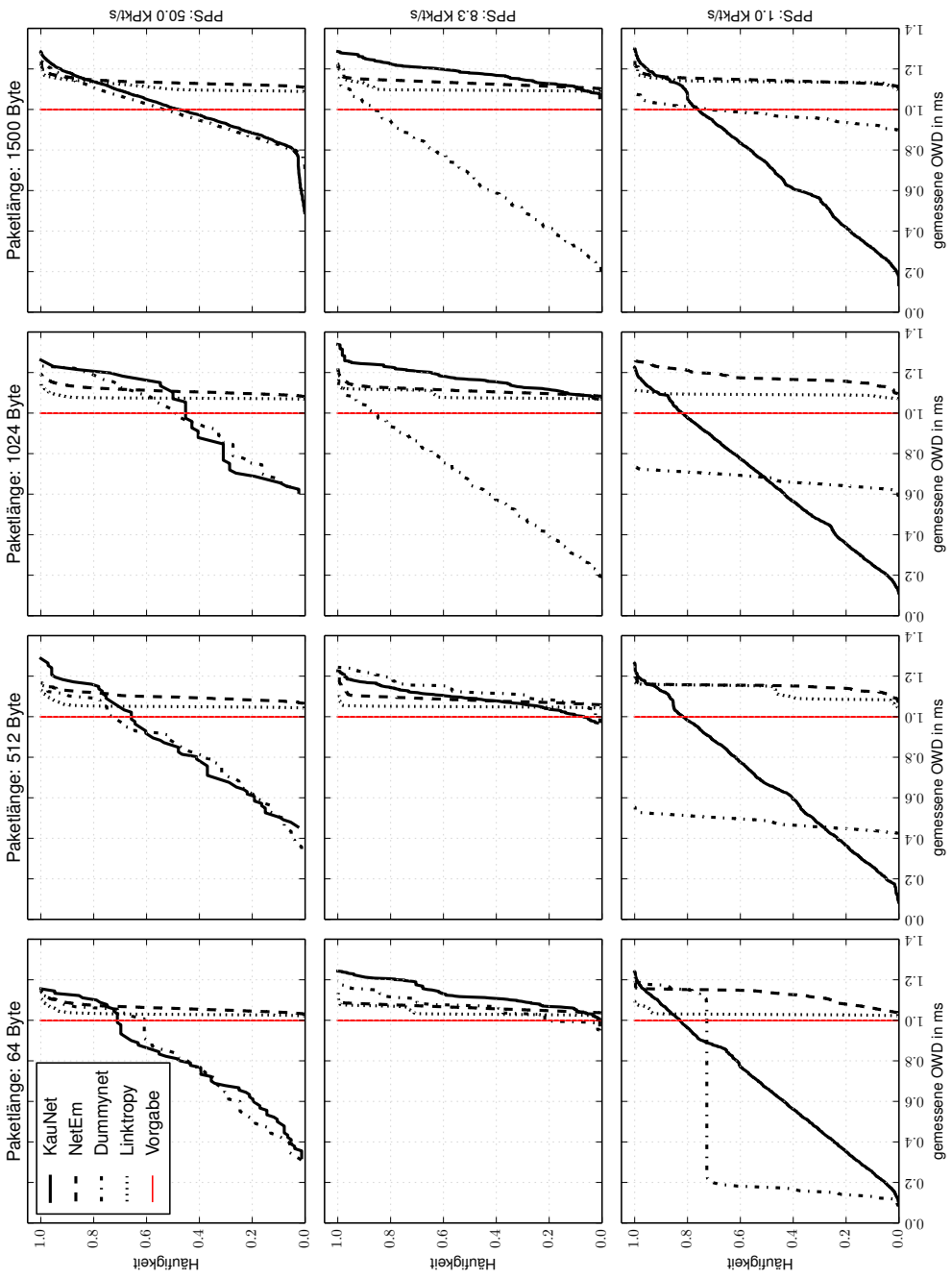


Abbildung A.4: Verzögerungen als Verteilungsfunktion des jeweiligen Emulators für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt. Vorgabe: 1 ms

A.7 Messung: Neuordnung

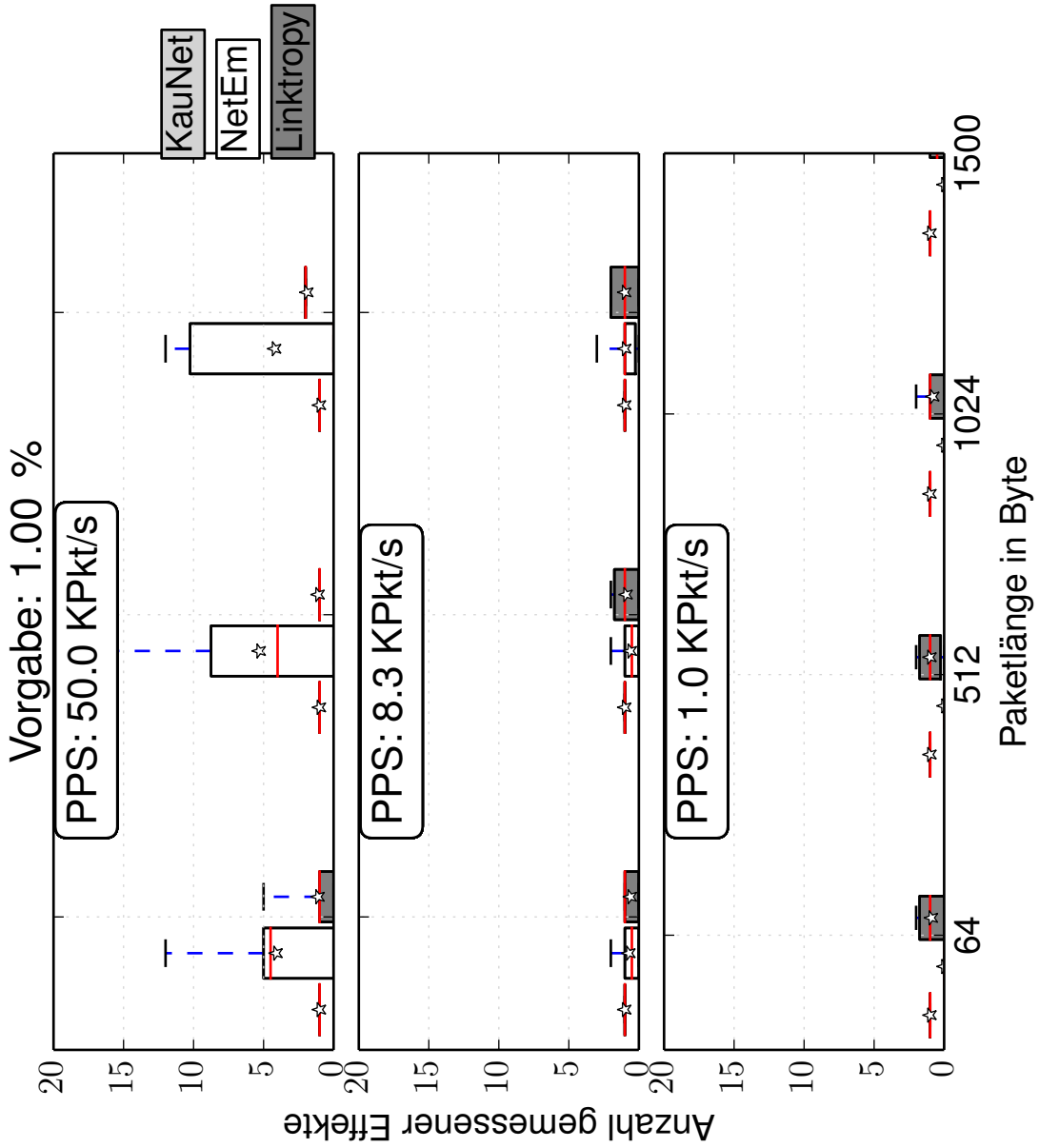


Abbildung A.5: Erfasste Neuordnungen für die verschiedenen Paketlängen und -raten bei einem einzelnen Pfadabschnitt. Vorgabe: 1 %

A.8 Erklärung zum Urheberrecht

Hiermit versichere ich, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet.

Dresden, 30. Januar 2014

Bibliografie

B.9 Literaturverzeichnis

[Aga05] Agarwal, Shilpi: „NetPath Manual“. Wisconsin Advanced Internet Laboratory, 2005 (siehe S. 43)

[AKZ99b] Almes, G. et al.: „RFC 2681: A round-trip delay metric for IPPM“. Techn. Ber. RFC 2681, september, 1999 (siehe S. 14)

[All+09] Allman, M. et al.: „RFC 5681: TCP congestion control“. Techn. Ber. 2009 (siehe S. 13)

[AP12] Alberto Dainotti, Alessio Botta and et al.: A tool for the generation of realistic network workload for emerging networking scenarios. In: *Computer Networks* 56 (2012) Nr. 15, S. 3531–3547 (siehe S. 37)

[Arl05] Arlos, Patrik: „On the quality of computer network measurements“. 2005 (siehe S. 46)

[Bro03] Broy, M.: „Kommunikations- und Informationstechnik 2010+3: Neue Trends und Entwicklungen in Technologie, Anwendungen und Sicherheit“. SecuMedia-Verlag, 2003. ISBN: 9783922746485 (siehe S. 1)

[Büs13] Büschel, P.: „Nora: Integrated Network Measurement Tool for End-to-End Connections“. 2013 (siehe S. 2, 9, 11, 13–15, 23, 61, 68, 69, 74, 84)

[CR10] Carbone, Marta et al.: Dummynet revisited. In: *SIGCOMM Comput. Commun. Rev.* 40 (Apr. 2010) Nr. 2, S. 12–20. ISSN: 0146-4833. DOI: 10.1145/1764873.1764876. URL: http://doi.acm.org/10.1145/1764873.1764876 (siehe S. 23, 28–30)

[GHB08] Garcia, Johan et al.: KauNet: A versatile and flexible emulation system. In: *Proceedings of the 5th Swedish National Computer Networking Workshop (SNCNW)*. 2008 (siehe S. 30)

[Gra+08] Grau, Andreas et al.: Time jails: A hybrid approach to scalable network emulation. In: *Principles of Advanced and Distributed Simulation, 2008. PADS'08. 22nd Workshop on*. IEEE. 2008, S. 7–14 (siehe S. 4)

[Hal+12] Hall, Tomas et al.: „Performance Evaluation of KauNet in Physical and Virtual Emulation Environments“. Techn. Ber. 2012:32. Karlstad University, Department of Computer Science, 2012, S. 49 (siehe S. 32, 33, 61, 62)

[Hem+05] Hemminger, Stephen et al.: Network emulation with NetEm. In: *Linux Conf Au*. Citeseer. 2005, S. 18–23 (siehe S. 36, 37)

[HK05] Haq, Furqan et al.: Simulation vs. emulation: Evaluating mobile ad hoc network routing protocols. In: *Proceedings of the International Workshop on Wireless Ad-hoc Networks (IWWAN 2005)*. 2005 (siehe S. 4)

[Hur+12] Hurtig, Per et al.: KauNet 2.0: Design and Usage. In: (2012) (siehe S. 31)

[IHL07] Ivanov, Svilen et al.: Experimental validation of the ns-2 wireless model using simulation, emulation, and real network. In: *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*. VDE. 2007, S. 1–12 (siehe S. vi, 4, 41, 42)

[Jur+11] Jurgelionis, Audrius et al.: An empirical study of netem network emulation functionalities. In: *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*. IEEE. 2011, S. 1–6 (siehe S. 37)

[KN08] Kalitay, Hemanta Kumar et al.: WANem 2.0 Wide Area Network Emulator Performance Engineering Research Centre. In: TATA Consultancy services. 2008 (siehe S. 39)

[KN11] Kalitay, Hemanta Kumar et al.: Designing wanem: A wide area network emulator tool. In: *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*. IEEE. 2011, S. 1–4 (siehe S. 38)

[Koh+00] Kohler, Eddie et al.: The Click modular router. In: *ACM Transactions on Computer Systems (TOCS)* 18 (2000) Nr. 3, S. 263–297 (siehe S. 42)

[Luc+03] Lucio, Gilberto Flores et al.: Opnet modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed. In: *WSEAS Transactions on Computers* 2 (2003) Nr. 3, S. 700–707 (siehe S. 3)

[McC+97] McCanne, Steven et al.: „Network simulator ns-2“. 1997 (siehe S. 35)

[MFA10] Minhas, Tahir Nawaz et al.: Quantification of packet delay variation through the coefficient of throughput variation. In: *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*. ACM. 2010, S. 336–340 (siehe S. 46, 81)

[Mil92] Mills, DL: RFC 1305: Network time protocol. In: *Specification, Implementation and Analysis* (1992) (siehe S. 63)

[Min+11] Minhas, Tahir Nawaz et al.: Evaluation of throughput performance of traffic shapers. In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*. IEEE. 2011, S. 1596–1600 (siehe S. 44, 48, 70)

[Nap08] Napoli, University of: Flyer2: D-ITG Distributed Internet Traffic Generator. In: (2008) (siehe S. 68)

[NK11] Nambiar, Manoj et al.: Design of a new algorithm for WAN disconnection emulation and implementing it in WANem. In: *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*. ACM. 2011, S. 376–381 (siehe S. 17, 39)

[NR09] Nussbaum, Lucas et al.: A comparative study of network link emulators. In: *Proceedings of the 2009 Spring Simulation Multiconference*. Society for Computer Simulation International. 2009, S. 85 (siehe S. 5, 44–46, 50, 61)

[PM06] Puljiz, Zrinka et al.: IMUNES based distributed network emulator. In: *Software in Telecommunications and Computer Networks, 2006. SoftCOM 2006. International Conference on*. IEEE. 2006, S. 198–203 (siehe S. 39, 40)

[PPM08] Puljiz, Zrinka et al.: Performance analysis of a decentralized network simulator based on IMUNES. In: *Performance Evaluation of Computer and Telecommunication Systems, 2008. SPECTS 2008. International Symposium on*. IEEE. 2008, S. 519–525 (siehe S. 40, 41)

[Riz97] Rizzo, Luigi: Dummynet: a simple approach to the evaluation of network protocols. In: *ACM SIGCOMM Computer Communication Review* 27 (1997) Nr. 1, S. 31–41 (siehe S. 26, 27)

[Sha+10] Shaikh, Junaid et al.: Evaluation of delay performance of traffic shapers. In: *Security and Communication Networks (IWSCN), 2010 2nd International Workshop on*. IEEE. 2010, S. 1–8 (siehe S. 44, 46–48, 50, 61, 62, 81)

[SU03] Simmonds, Rob et al.: Towards scalable network emulation. In: *Computer Communications* 26 (2003) Nr. 3, S. 264–277 (siehe S. 4, 5)

[Tec11] Technologies, Apposite: „Linktropy 7500 pro Hardware Guide“. Bd. 5. apposite, 2011 (siehe S. 26)

[Tec13] Technologies, Apposite: „Linktropy Wan Emulator User's Guide“. Bd. 4. apposite, 2013 (siehe S. 26)

[Uni12] University, of Zagreb: „IMUNES Manual“. University of Zagreb, 2012 (siehe S. 40, 42)

[Yoc+02] Yocum, Ken et al.: Scalability and accuracy in a large-scale network emulator. In: *ACM SIGCOMM Computer Communication Review* 32 (2002) Nr. 3, S. 28–28 (siehe S. 33–35)

[ZM04] Zec, Marko et al.: Operating system support for integrated network emulation in imunes. In: *1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure (OASIS)*. 2004, S. 3–12 (siehe S. 39, 40)

B.10 Quellenverzeichnis

[12] „The Tcpdump and Libpcap Web site“. 2012. URL: http://www.tcpdump.org (besucht am 23. 12. 2012) (siehe S. 32)

[13a] „Apposite's Linktropy and Netropy WAN emulators simulate bandwidth, delay, loss, and other network impairments." 2013. URL: http://www.apposite-tech.com/index.html (besucht am 13. 05. 2013) (siehe S. 23)

[13b] „CT503-MIX High-Speed LANforge-FIRE Traffic Generator". 2013. URL: http://www.candelatech.com/ct503-MIX_product.php (besucht am 02. 12. 2013) (siehe S. 68)

[13c] „Emulab - Network Emulation Testbed Home". 2013. URL: http://www.emulab.net/ (besucht am 07. 09. 2013) (siehe S. 23)

[13d] „FreeBSD-Jails". 2013. URL: http://www.freebsd.org/doc/de_DE.ISO8859-1/books/handbook/jails.html (besucht am 06. 09. 2013) (siehe S. 19, 39)

[13e] „Google Scholar". 2013. URL: http://scholar.google.de/ (besucht am 07. 09. 2013) (siehe S. 23, 24)

[13f] „Google Trendanalyse bzgl. Dummynet, Netem, Modelnet, NetPath, Emulab". 2013. URL: http://www.google.de/trends/explore?q=Dummynet#q=Dummynet,%20Netem,%20Modelnet,%20NetPath,%20Emulab&cmpt=q (besucht am 07. 09. 2013) (siehe S. 24)

[13g] „Grid'5000, a scientific instrument designed to support experiment-driven research in all areas of computer science related to parallel, large-scale or distributed computing and networking". 2013. URL: https://www.grid5000.fr/ (besucht am 26. 11. 2013) (siehe S. 61)

[13h] „iperf". 2013. URL: http://iperf.fr/ (besucht am 02. 12. 2013) (siehe S. 68)

[13i] „Ixia Anue 3500". 2013. URL: http://www.ixiacom.com/products/network_test/load_modules/anue_3500/ (besucht am 02. 12. 2013) (siehe S. 68)

[13j] „KAUNET - DETERMINISTIC NETWORK EMULATION". 2013. URL: http://www.kau.se/en/kaunet (besucht am 07. 09. 2013) (siehe S. 23)

[13k] „Kernel Probes (Kprobes)". 2013. URL: https://www.kernel.org/doc/Documentation/kprobes.txt (besucht am 09. 09. 2013) (siehe S. 37)

[13l] „Libpcap". 2013. URL: http://www.tcpdump.org/ (besucht am 13. 02. 2013) (siehe S. 67)

[13m] „Modeling Topology of Large Internetworks". 2013. URL: http://www.cc.gatech.edu/projects/gtitm/ (besucht am 07. 09. 2013) (siehe S. 34)

[13n] „ModelNet Systems and Networking". 2013. URL: http://modelnet.ucsd.edu/ (besucht am 07. 09. 2013) (siehe S. 23)

[13o] „Multi-Generator (MGEN)". 2013. URL: http://cs.itd.nrl.navy.mil/work/mgen/ (besucht am 18. 09. 2012) (siehe S. 32)

[13p] „netem GUI". 2013. URL: http://phpnetemgui.sourceforge.net/ (besucht am 07. 09. 2013) (siehe S. 36)

[13q] „netem Network Emulation“. 2013. URL: http://www.linuxfoundation.org/collaborate/workgroups/networking/netem (besucht am 07. 09. 2013) (siehe S. 24, 38)

[13r] „NetPath Präsentation“. 2013. URL: http://pages.cs.wisc.edu/~jsommers/pubs/netpath_slides.pdf (besucht am 09. 09. 2013) (siehe S. 24)

[13s] „NISTNet Home Page“. 2013. URL: http://www-x.antd.nist.gov/nistnet/ (besucht am 07. 09. 2013) (siehe S. 24)

[13t] „Ostinato, Packet/Traffic Generator and Analyzer“. 2013. URL: https://code.google.com/p/ostinato/ (besucht am 02. 12. 2013) (siehe S. 68)

[13u] „Other Internet Traffic Generators“. 2013. URL: http://traffic.comics.unina.it/software/ITG/link.php (besucht am 02. 12. 2013) (siehe S. 68)

[13v] „Packet Crafter Apps“. 2013. URL: http://www.packet-craft.net/Apps/ (besucht am 02. 12. 2013) (siehe S. 68)

[13w] „The dummynet project“. 2013. URL: http://info.iet.unipi.it/~luigi/dummynet/ (besucht am 07. 09. 2013) (siehe S. 23, 30)

[13x] „The Genius of DAG“. 2013. URL: http://www.emulex.com/products/network-visibility-products/the-genius-of-dag/features/ (besucht am 06. 10. 2013) (siehe S. 47)

[13y] „Traffic Control HOWTO: 4. Components of Linux Traffic Control“. 2013. URL: http://tldp.org/HOWTO/Traffic-Control-HOWTO/components.html (besucht am 09. 09. 2013) (siehe S. 36)

[13z] „WANem The Wide Area Network emulator“. 2013. URL: http://wanem.sourceforge.net/ (besucht am 09. 09. 2013) (siehe S. 24)

[13aa] „winpcap“. 2013. URL: http://www.winpcap.org/ (besucht am 13. 02. 2013) (siehe S. 87)

[AKZ99a] Almes, G. et al.: „RFC 2679: A one-way delay metric for IPPM“. Techn. Ber. 1999, S. 1–20 (siehe S. 14)

[Mic04] Michaut, F.: „“QoSMet, a Quality of Service Measurement Tool“. 2004. URL: http://michaut.valerie.free.fr/qosmet (besucht am 23. 12. 2012) (siehe S. 37)

[Pra+03] Prasad, R. et al.: Bandwidth estimation: metrics, measurement techniques, and tools. In: Bd. 17. 6. IEEE, 2003, S. 27–35 (siehe S. 11, 12)